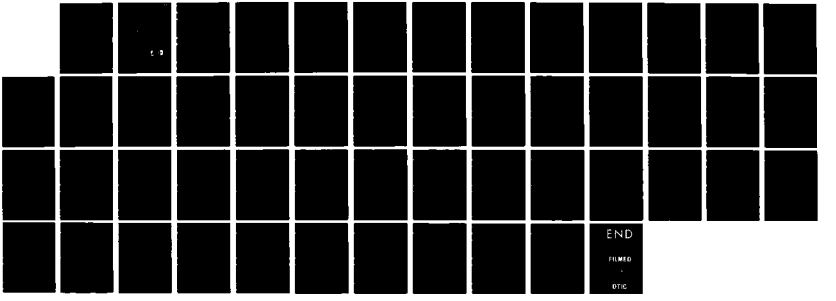
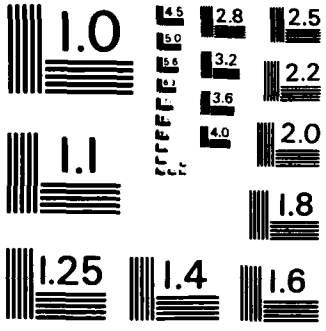
AD-A161 326 A UNIFIED APPROACH TO THE ANALYSIS AND SYNTHESIS OF SYSTOLIC ARRAYS(U) ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY GROUP S W HORNICK APR 85 ACT-56 1/1

UNCLASSIFIED N00014-84-C-0149 F/G 9/2 NL





REPORT ACT-56

APRIL 1985

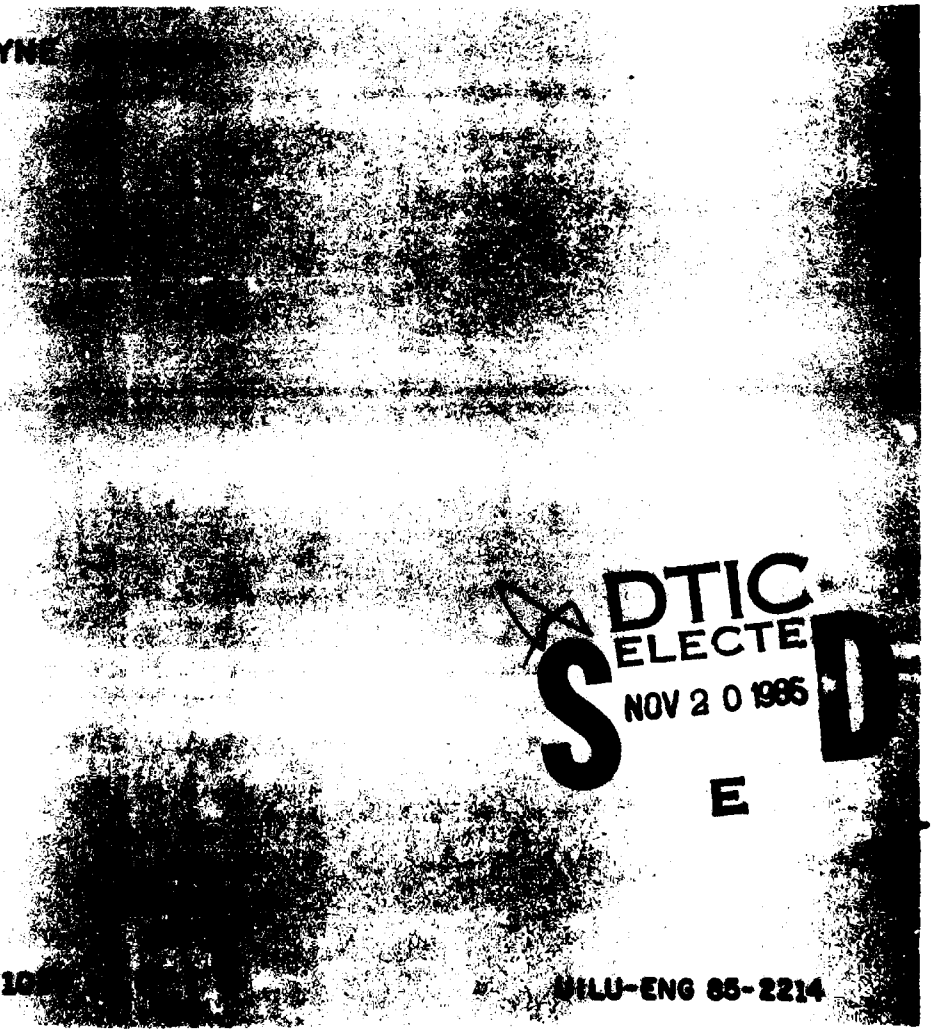
(12)





# A UNIFIED APPROACH TO THE ANALYSIS AND SYNTHESIS OF SYSTOLIC ARRAYS

SCOT WAYNE







REPORT R-10

UILU-ENG 85-2214

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION: Unclassified | 1b. RESTRICTIVE MARKINGS: None |
| 2a. SECURITY CLASSIFICATION AUTHORITY: N/A | 3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S): | 5. MONITORING ORGANIZATION REPORT NUMBER(S): N/A |
| 6a. NAME OF PERFORMING ORGANIZATION: Coordinated Science Laboratory, University of Illinois; 6b. OFFICE SYMBOL (If applicable): N/A | 7a. NAME OF MONITORING ORGANIZATION: Joint Services Electronics Program |
| 6c. ADDRESS (City, State and ZIP Code): 1101 W. Springfield Ave. Urbana, IL 61801 | 7b. ADDRESS (City, State and ZIP Code): 800 N. Quincy St. Arlington, VA 22217 |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Schlumberger Fellowship and Joint Services Electronics Program; 8b. OFFICE SYMBOL (If applicable): N/A | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: Contract N00014-84-C-0149 |
| 8c. ADDRESS (City, State and ZIP Code): 800 N. Quincy St. Arlington, VA 22217 | 10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO.: N/A; PROJECT NO.: N/A; TASK NO.: N/A; WORK UNIT NO.: N/A |
| 11. TITLE (Include Security Classification): A Unified Approach to the Analysis and Synthesis of Systolic Arrays | |
| 12. PERSONAL AUTHOR(S): Hornick, Scot Wayne | |
| 13a. TYPE OF REPORT: R-1039 ACT-56;UILU-ENG 85-2214; 13b. TIME COVERED: FROM _____ TO _____ | 14. DATE OF REPORT (Yr., Mo., Day): April 1985; 15. PAGE COUNT: 45 |
| 16. SUPPLEMENTARY NOTATION: N/A | |
| 17. COSATI CODES — FIELD: ; GROUP: ; SUB. GR.: | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Systolic arrays, ~~VLSI design~~, parallel algorithms, matrix computations; discrete convolution |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this thesis, we take the first steps toward the development of a theoretical framework to unify the analysis and synthesis of systolic networks. We describe a class of transformations on systolic networks that alter the topology of a network while preserving the timing of its computations. These transformations may be used to demonstrate the equivalence of two existing systolic designs or to obtain a new design from an existing one, according to particular design specifications. We present our model of systolic network and then identify the parameters that we use to characterize one. Next, we prove the correctness of two types of transformations on these parameters. We show how these transformations can account for different processor types and multiple processor states. Finally, we demonstrate these transformations and characterize those that avoid the phenomenon of deadlock.

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐ | 21. ABSTRACT SECURITY CLASSIFICATION: Unclassified |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE NUMBER (Include Area Code); 22c. OFFICE SYMBOL: NONE |

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

# A UNIFIED APPROACH TO THE ANALYSIS AND SYNTHESIS OF SYSTOLIC ARRAYS

BY

SCOT WAYNE HORNICK

B.S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1985



Urbana, Illinois

# ACKNOWLEDGEMENTS

I gratefully acknowledge the patient guidance and perceptive insight of my thesis advisor, Professor Franco P. Preparata. I have also profited from several discussions with my friend and colleague, Gianfranco Bilardi. In addition, I am indebted to Julie Overholt for typing the manuscript and formatting it for phototypesetting.

The work reported in this thesis has been supported in part by a Schlumberger Fellowship and by the Joint Services Electronics Program under contract N00014-84-C-0149.

Finally, I want to express my gratitude to my wife, Lori, for proofreading the thesis, for helping to draw the figures, and for supporting and encouraging my work.

# TABLE OF CONTENTS


| | Page |
|---|---|
| 1. INTRODUCTION | 1 |
| 2. MODEL OF A SYSTOLIC NETWORK | 3 |
| 2.1. MODEL OF PROCESSING ELEMENTS | 3 |
| 2.2. MODEL OF DATA FLOWS | 4 |
| 3. TRANSFORMATION OF DATA FLOWS | 7 |
| 3.1. MATHEMATICAL DEFINITION | 7 |
| 3.2. CANONICAL REPRESENTATION | 10 |
| 3.3. TRANSFORMATION OF STATE FLOWS | 10 |
| 3.4. EXAMPLES OF TRANSFORMATIONS | 12 |
| 3.5. CROSSINGS IN TRANSFORMED NETWORKS | 34 |
| 4. CONCLUSION | 43 |
| REFERENCES | 44 |

# 1. INTRODUCTION

Recent advances in semiconductor device fabrication technology have made possible the realization of increasingly complex digital integrated circuits. VLSI computation theory addresses the problem of efficiently using this chip complexity (as measured by area) in order to decrease computation time. The search for efficient use of area and time resources has borne certain classes of architectures that are repeatedly utilized.

One such class is that of systolic arrays. Systolic arrays have been described by H. T. Kung and C. E. Leiserson [8]:

> "A systolic system is a network of processors which rhythmically compute and pass data through the system.... Every processor regularly pumps data in and out, each time performing some short computation, so that a regular flow of data is maintained in the network."

Systolic networks exhibit regular and modular layouts. In addition, interprocessor connections are bounded in number and localized in space. These features make systolic architectures particularly well suited to the planar format imposed by VLSI technology. Systolic architectures also interface with conventional computer memories in a natural and efficient way. A number of systolic arrays have been proposed by a number of authors [1,2,3,4,6,7,8,10,11]. These arrays seem most promising in certain numerical computations; proposed applications include: discrete convolution, matrix multiplication, LU and other matrix decompositions, triangular system solution, and other related computations.

The systolic arrays that have been proposed thus far share many common features. However, there has been little theory unifying these designs, and most have been presented ad hoc, without detailed analysis. Here, we take the first steps toward the development of a theoretical framework to unify the analysis and synthesis of systolic networks. We describe a class of transformations on systolic networks that alter the topology of a network while preserving the timing of its computations. These transformations may be used to demonstrate the equivalence of two existing systolic designs or to obtain a new design from an existing one.

This thesis is organized as follows. In the second chapter, we discuss our model of a systolic network and identify the parameters that we will use to characterize one. In the third chapter, we present

the main result of the thesis, two theorems that allow us to transform the parameters of a systolic network while preserving the nature of the computation that it performs. We demonstrate these transformations and characterize those that yield "crossing-free" systolic networks. Finally, in the fourth chapter, we present our conclusions and suggestions for further research.

# 2. MODEL OF A SYSTOLIC NETWORK

## 2.1. MODEL OF PROCESSING ELEMENTS

A *systolic network* can be viewed as a collection of processing elements (PEs) located at vertices of a multidimensional uniform grid. Each of the PEs can be partitioned (Figure 1) into a *control machine*, M, and a *computation machine*, N. Both of these can be considered finite state machines; however, references to the state of a PE will actually apply only to the state of the control machine. The state of the computation machine is the contents of its data registers and will be considered later.

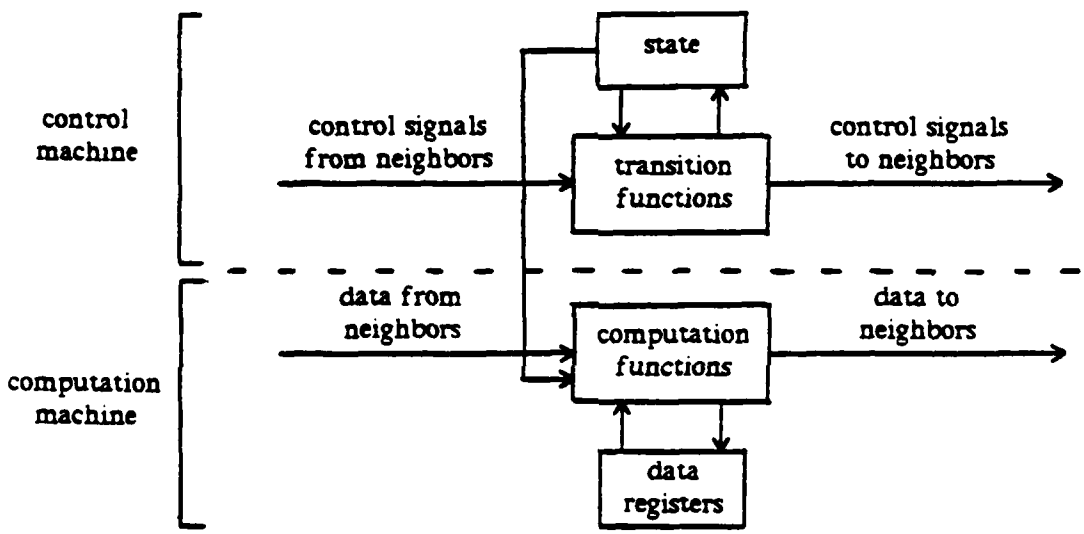



Figure 1. General model of a systolic PE

The control machine effects the correct state transitions for the PE, transmits the appropriate control signals to neighboring PEs, and controls the functions computed by the computation machine. In short, the control machine performs the synchronization and sequencing activities necessary to coordinate the operation of the systolic network. The computation machine, on the other hand, operates on the data flowing through the network. It performs the arithmetic involved in computing the actual output of the network. Using the formalism of finite state machines, we have

$$s' = \delta_M(s,C_I),\ C_O = \lambda_M(s,C_I),\ R' = \delta_N(R,D_I,s),\ D_O = \lambda_N(R,D_I,s),$$

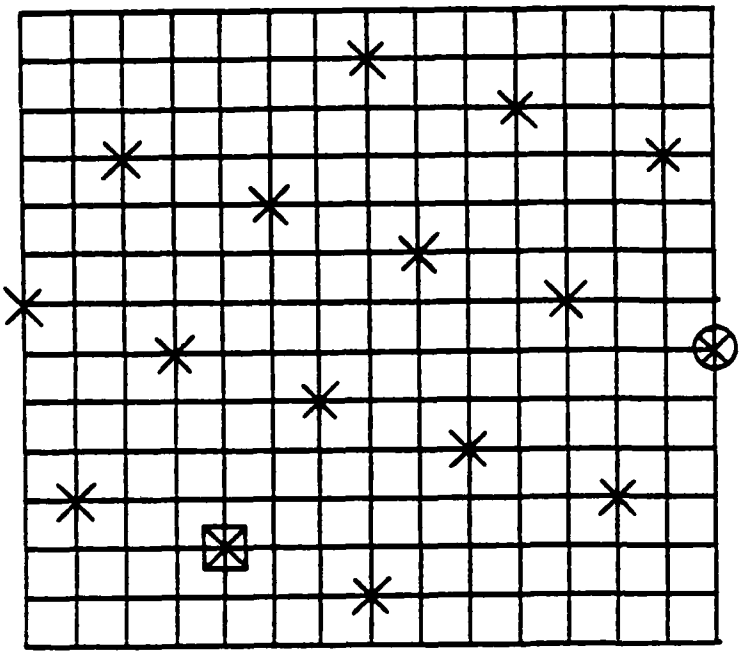
where s, R, C, and D are the PE state, the PE register contents, the control signals, and the data,

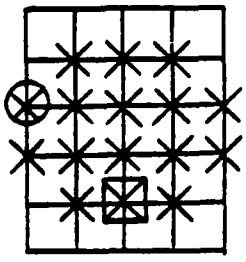
respectively. Also, the subscripts I and O denote input and output, respectively, and $\delta$ and $\lambda$ are respectively the state transition and output functions.

The PEs in a systolic network are selected from a set of possible module types. Some systolic networks utilize only one type of module, while others utilize as many as four types. In all cases, however, the number of module types is independent of the network size and is, instead, determined by the type of computation performed by the network. Each module type is characterized by a *module description*. The *module description* specifies the states in which a PE of this module type can be. Each of these states, in turn, is characterized by a *state description*. A *state description* consists of a collection of assignment statements and control statements to be executed by a PE in this state at the end of every clock cycle. The assignment statements are written in a register transfer language and dictate the operation of the computation machine; they indicate which input ports or registers serve as sources of operands, which operations are performed on the operands, and which output ports or registers serve as destinations for the results. The control statements dictate the operation of the control machine; they indicate state transitions to be executed by the PE or its neighbors. A state transition to be executed by a neighbor must be initiated by the PE via control signals.

## 2.2. MODEL OF DATA FLOWS

The locations of the PEs in the uniform grid are constrained in two ways. First, consider the subset of PEs belonging to a particular module type. We will require that the set of positions occupied by these PEs forms a *lattice*. A set, P, of points on a uniform grid is a *lattice* if $P = \{p \mid p = Lg + d, g \in G\}$, where G is the set of unit-grid points contained in a closed convex domain (*G is referred to briefly as a convex grid set*), L (the *distortion matrix*) is a matrix of rational numbers mapping each unit-grid point to a uniform-grid point, and d is a fixed grid point (the *origin*); i.e., P must be the result of an affine transformation on a convex grid set, G, (Figure 2). The notation p is used here to represent a d-component vector $[p_1 \ p_2 \ \cdots \ p_d]^T$, where d is the dimension of the uniform grid in which the PEs are located.





G

$$L = \begin{bmatrix} -3 & 2 \\ 1 & 3 \end{bmatrix}$$

Figure 2. Example of a lattice in two dimensions, the underlying convex grid set (G), and the distortion matrix (L)

Secondly, consider a PE at position **p**. If this PE receives input data directly (in unit time) from another PE at position $\mathbf{p}-\mathbf{v}$ ($\mathbf{v}\neq 0$), then, in order to maintain a regular flow of data, we will require that it be able to transmit output data directly (in unit time) to a PE at position $\mathbf{p}+\mathbf{v}$ (except, possibly, for boundary PEs). Thus, there must exist a PE at $\mathbf{p}+\mathbf{v}$, a directed edge for data communication with terminals $(\mathbf{p}-\mathbf{v},\mathbf{p})$, and one with terminals $(\mathbf{p},\mathbf{p}+\mathbf{v})$.

If we associate all the data that either flow along communication edges with a particular length and orientation or reside in a particular set of computation machine registers, this will define a *data flow*. Formally, a *data flow* is a pair $C = \langle A,\phi \rangle$ defined as follows:

1. A is a set of data (inputs, outputs, or intermediate results of a systolic computation);
2. $\mathbb{N}$, the set of natural integers, corresponds to the set of instants of discrete time ($t \in \mathbb{N}$ is the index of the time unit);
3. $\phi : A \times \mathbb{N} \rightarrow U$ is an injective function of the data set and discrete time that maps, at any given t, each of the elements of A to a position on the uniform grid, U. Denote by $\phi(A,t)$ the range

of $\phi$ at time t. Function $\phi$ is further constrained as follows:

a. $\frac{\phi(a,t+k) - \phi(a,t)}{k} = \mathbf{v}$ for all $a \in A$ and all $k,t \in \mathbb{N}$

(i.e., all elements of A move with the same constant velocity, $\mathbf{v}$).

b. $\phi(A,t)$ must be a lattice for any integer t.

Combining (3a) and (3b), we obtain $\phi(A,t) = \{p \mid p = Lg + d + t\mathbf{v}, g \in G\}$. Thus, we can define an injective, time independent function, $\gamma$, which maps elements of A to a position in G, a given convex grid set:

$$\gamma : A \rightarrow G.$$

Therefore, $\phi(a,t) = L\gamma(a) + d + t\mathbf{v}$.

# 3. TRANSFORMATION OF DATA FLOWS

## 3.1. MATHEMATICAL DEFINITION

In this section, we define a class of transformations on data flows that alter the topology of a systolic network without changing the timing of its data interaction. Other authors have suggested similar, but somewhat different transformations. In particular, Leiserson and Saxe have proven a "Systolic Conversion Theorem" that converts nonsystolic networks to systolic networks [9]. However, their conversion is effected by retiming the computations of a network without changing the topology of the underlying communication graph. Cappello and Steiglitz have also suggested transformations to unify the design of systolic networks [5]. They have described linear transformations of space-time that are capable of altering both network topology and computation timing. As we shall explain later, the transformations described by Cappello and Steiglitz are especially similar to ours, which are characterized by the two following theorems.

Theorem 1:

A constant vector, $\mathbf{u}$, may be added to all data flow velocities without altering the data flow origins, the distortion matrices, $\gamma$, or the timing of the computations.

Proof: Consider two arbitrary elements of two different data flows, $x \in X$ and $y \in Y$, that must interact at some time $t$. This constrains X and Y to satisfy $\phi(x,t) = \phi(y,t)$. If $L_x$, $d_x$, and $\mathbf{v}_x$ denote the distortion matrix, the origin, and the velocity of data flow X, respectively, and $L_y$, $d_y$, and $\mathbf{v}_y$ denote the corresponding entities of data flow Y, we can write this constraint as:

$$L_x\gamma_x(x) + d_x + t\mathbf{v}_x = L_y\gamma_y(y) + d_y + t\mathbf{v}_y.$$

We now show that if $\mathbf{v}_x'$ and $\mathbf{v}_y'$ are the transformed velocities of the X and Y data flows, respectively, the above constraint is still satisfied.

Let $\mathbf{v}_x' = \mathbf{v}_x + \mathbf{u}$, $\mathbf{v}_y' = \mathbf{v}_y + \mathbf{u}$:

$$L_x\gamma_x(x) + d_x + tv_x = L_y\gamma_y(y) + d_y + tv_y$$

$$L_x\gamma_x(x) + d_x + tv_x + tu = L_y\gamma_y(y) + d_y + tv_y + tu$$

$$L_x\gamma_x(x) + d_x + t(v_x + u) = L_y\gamma_y(y) + d_y + t(v_y + u)$$

$$L_x\gamma_x(x) + d_x + tv_x' = L_y\gamma_y(y) + d_y + tv_y'$$

Furthermore, since this transformation is invertible, no additional interactions are introduced by it, i.e., there is a one-to-one correspondence between the data interactions in the original network and the data interactions in the resultant network. □

**Theorem 2:**

All of the data flow velocities, origins, and distortion matrices may be multiplied by a nonsingular matrix, M, without altering $\gamma$ or the timing of the computations.

**Proof:** Again, we consider two arbitrary interacting elements and show that if $v_x'$ and $v_y'$ are the transformed velocities, $d_x'$ and $d_y'$ are the transformed origins, and $L_x'$ and $L_y'$ are the transformed distortion matrices of the X and Y data flows, respectively, the positional constraint is still satisfied.

Let $v_x' = Mv_x$, $v_y' = Mv_y$, $d_x' = Md_x$, $d_y' = Md_y$, $L_x' = ML_x$, $L_y' = ML_y$:

$$L_x\gamma_x(x) + d_x + tv_x = L_y\gamma_y(y) + d_y + tv_y$$

$$M(L_x\gamma_x(x) + d_x + tv_x) = M(L_y\gamma_y(y) + d_y + tv_y)$$

$$ML_x\gamma_x(x) + Md_x + tMv_x = ML_y\gamma_y(y) + Md_y + tMv_y$$

$$L_x'\gamma_x(x) + d_x' + tv_x' = L_y'\gamma_y(y) + d_y' + tv_y'$$

Since M is nonsingular, this transformation is also invertible, and, again, there is a one-to-one correspondence between the data interactions in the original network and the data interactions in the resultant network. □

These two theorems provide a simple, yet powerful set of rules for transforming systolic networks. Networks that are equivalent under these transformations are said to be *affinely equivalent* or simply *equivalent*. Networks that are equivalent under transformations of the second type alone (those

described by Theorem 2) are said to be *linearly equivalent*. In general, these affine transformations result in topological changes in the network, but, if we restrict them to be linear, they result only in "conformal" changes in the network, i.e., dilation, contraction, rotation, or reflection. The intermediate results of such transformations are arbitrary, but the final result must represent a valid set of data flows; in particular, each of the velocities and distortion matrices must consist of rational elements. In subsequent portions of this thesis, we will tacitly ignore the data flow origins since these parameters can easily be obtained after transformation through initial condition considerations.

As was noted previously, the affine transformation of data flow parameters is very similar to the linear transformation of space-time as described by Cappello and Steiglitz. In fact, in cases where both can be applied, our affine transformations are a special case of the Cappello-Steiglitz transformations. Specifically, Theorem 1 describes transformations that Cappello and Steiglitz would represent by the matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix},$$

while Theorem 2 describes transformations that they would represent by the matrix

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

(The last coordinate is taken to be time.) We feel, however, that this loss of generality is compensated by the following considerations. First, the affine transformations give the designer more of a "kinematic" intuition of the design process and are simpler to use if one is given a systolic network a priori. The Cappello-Steiglitz transformations, on the other hand, require the geometric description of an algorithm. Second, the set of systolic networks is closed under affine transformation. However, Cappello-Steiglitz transformations may yield designs that are unrealistic in the VLSI model of computation, e.g., designs with unbounded fan-in or fan-out. (This is necessary, of course, in contexts where such designs are to be studied.) Finally, as we shall see, affine transformations may also be used to derive the module descriptions of a transformed network. This task becomes nontrivial when dealing

with systolic networks having multiple module types or a module type with multiple states.

## 3.2. CANONICAL REPRESENTATION

Since affine transformations can yield a number of equivalent systolic networks, it will be useful to distinguish a *canonical network*. The typical systolic computation is defined in terms of an operation, a set of one or more operands, a result, and, possibly, a set of side conditions. For a particular computation, a systolic network has inputs and outputs corresponding in some way to the operands and the result. This correspondence is unrestricted, i.e., inputs need not correspond to operands, and outputs need not correspond to the result. The systolic network computes the outputs (whether they are results or operands) so that the result is consistent with the operation on the operands, subject to any existing side conditions.

With this in mind, we define the *canonical network* as one in which the result data flow has zero velocity and an identity distortion matrix. All systolic networks can then be represented as a two step transformation of a canonical design: first, we add a vector to all data flow velocities of the canonical design (according to Theorem 1); second, we multiply all data flow velocities and distortion matrices by a nonsingular matrix (according to Theorem 2). This representation will be called the *canonical representation* of the network. Thus, a set of networks that share the same canonical network is an affine equivalence class, and a set of networks that share the same canonical network and the same first step of their canonical representations is a linear equivalence class.

## 3.3. TRANSFORMATION OF STATE FLOWS

At first, one might suspect that the module descriptions of a systolic network resulting from an affine transformation cannot be recovered easily from the module descriptions of the original network. Conceptually, though, it is quite simple. Let us replace each of the PEs of the original array with an *emulator*, E, that, when given an input and a state chosen from the union of all possible states of all module types, computes the output generated by a PE in the given state upon receipt of that input. The emulator is essentially a computation machine general enough to compute any function of any module

type, and the current state is simply one of the inputs to the emulator (Figure 3).

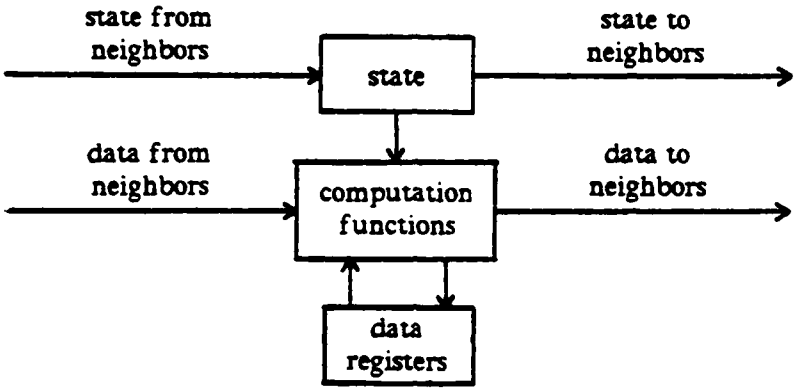



Figure 3. General model of a systolic emulator

The equations for the emulator are $R' = \delta_E(R,D_I,s)$ and $D_O = \lambda_E(R,D_I,s)$. The PE register contents (R) and the input/output data ($D_I$ and $D_O$) are data flow elements. (Register contents are elements of data flows with zero velocity.) Collectively, the states can be thought of as another data flow, which we will refer to as the *state flow*. This state flow can be transformed along with the other data flows. After the transformation, the peculiarities of the new state flow may suggest new module types that may be used to replace the emulators with hard-wired state transition control machines.

In order to carry out a transformation on the state flow, it is first necessary to have the module descriptions in an appropriate format. First, we assign a distinct label to each state of every type of module. This collection of labeled states forms the state alphabet from which elements of the state flow are selected. Second, we modify the state descriptions by removing all control statements and replacing input/output port and register designations in the assignment statements with the appropriate data flow labels. Control statements can be eliminated since the state flow now provides state transition information to module emulators, while port and register relabeling is necessary since data flow velocities are not preserved under affine transformation. For instance, data flowing from west to east in the original network may, after a suitable affine transformation, flow from south to north in the resulting network. Similarly, a data flow that is stationary, i.e., contained in module registers, may move from south to north after transformation. Thus, references to register names, east and west ports, or north

and south ports become meaningless after transformation. Any ambiguity in reference to input/output ports or registers by data flow label will be resolved in the following manner: a data flow label on the right-hand side of an assignment arrow will refer to the port through which elements of this data flow enter the module or, if the data flow is stationary, the register in which elements of it are contained; a data flow label on the left-hand side of an assignment arrow will refer to the port through which elements of this data flow exit the module or, if the data flow is stationary, the register in which elements of it are contained.

Abstracting the state flow from the control statements executed by the PEs is basically a problem in the theory of cellular automata. We know of no formalized or algorithmic approach to this problem; however, most of the instances encountered with systolic processing require only a limited effort and can be determined by inspection. The parameters to be determined are the velocity, a suitable convex grid set, and a distortion matrix. Once these are determined, they can be transformed along with the parameters of the data flows according to rules of Theorems 1 and 2.

## 3.4. EXAMPLES OF TRANSFORMATIONS

### 3.4.1. DISCRETE OPEN CONVOLUTION

Let us consider the problem of discrete open convolution, which is stated simply as follows: given a sequence of weights, W, and a sequence of inputs, X, compute the result sequence, $Y = W * X$, where the operation "*" is defined as $Y[i] = \sum_j W[j] \times X[i-j]$. Usually, W, X, or both have finite length, so the summation has finite bounds.

Kung has catalogued a family of linear systolic networks for discrete convolution in [7]. He refers to these as "(pure-) systolic" convolution arrays to distinguish them from "(semi-) systolic" arrays in which global fan-in or fan-out is necessary. Kung, however, has used a nonstandard definition of convolution where $Y[i] = \sum_j W[j] \times X[i+j-1]$. We prefer to adhere to the conventional definition of convolution since it preserves the symmetry (commutativity) of the operation, i.e., $W * X = X * W$. Clearly, convolution in the sense of Kung, which is conventionally referred to as "correlation," is

equivalent to convolving X delayed one time unit and W reversed in time. In other words, if Y is the convolution of X and W in the sense of Kung, then $Y = U * V$, where $U[k] = W[-k]$ and $V[k] = X[k-1]$. In this thesis, future references to the designs of Kung will incorporate the modifications necessary to compute the conventional convolution.

One possible canonical convolution network is that labeled R1 (results stay, inputs and weights move in opposite directions) by Kung (Figure 4). This network is essentially a pipelined implementation of the defining equation of convolution. Since the network is a linear array, the unit grid mentioned above reduces to the integer line. Similarly, the velocities and distortion matrices of the data flows reduce to rational scalars.

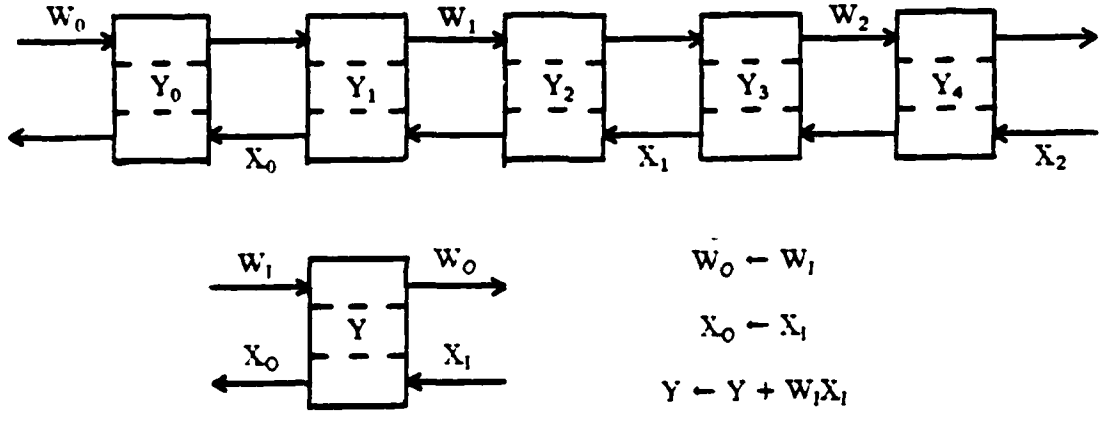



Figure 4. General structure of the canonical convolver (R1) and functional description of modules

There are three data flows in this network. The elements of W comprise an eastward data flow, the parameters of which will be subscripted with w. The elements of X comprise a westward data flow; parameters associated with it will be subscripted with x. The third data flow is stationary and consists of the elements of Y, which are contained in the registers of the PEs; parameters associated with this data flow will be subscripted with y. As a convention, we will define $\gamma$ for all sequences (and vectors) such that $\gamma(a[i]) = i$. We then obtain these values for the data flow parameters:

$$\text{canonical(R1)} \quad \begin{cases} v_w = 1, \; v_x = -1, \; v_y = 0, \\ L_w = 2, \; L_x = 2, \; L_y = 1. \end{cases}$$

If a network in which the weights are stationary is desired, we could simply subtract $v_w$ from all the velocities of the data flows in the canonical design. The resulting data flow velocities are (distortion matrices remain unchanged)

$$-1 \mid v_w = 0, v_x = -2, v_y = -1.$$

In this network (Figure 5), the weights are indeed stationary and inputs move in the same direction as the results, but at twice the speed. Kung mentions this network as a "dual" of design W2, although it now seems more closely related to R1. Transformation of the module descriptions is trivial since the only PEs in which computation occurs are inner product step processors.

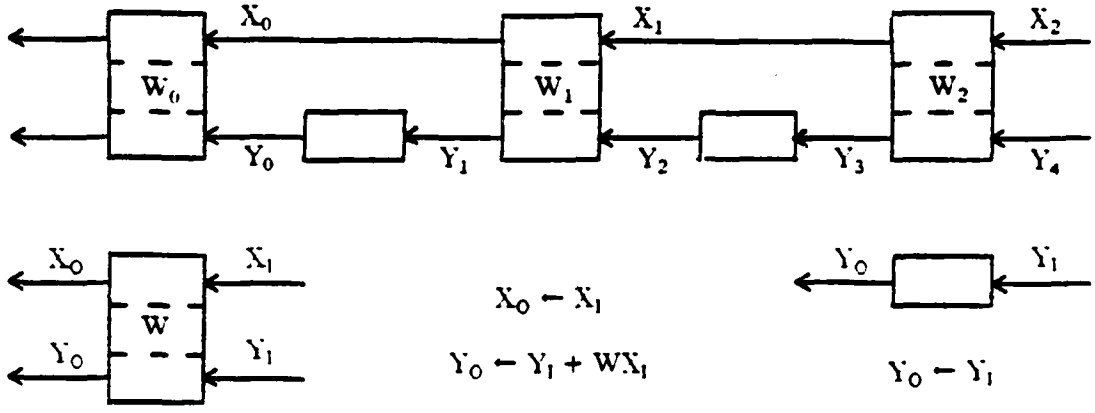



Figure 5. General structure of the -1 convolver and functional description of modules

A network with stationary inputs can be obtained by subtracting $v_x$ from the data flow velocities of the canonical network. The data flow velocities are then

$$+1 \mid v_w = 2, v_x = 0, v_y = 1.$$

In this network (Figure 6), which is not catalogued by Kung, the weights move in the same direction as the results, but at twice the speed.

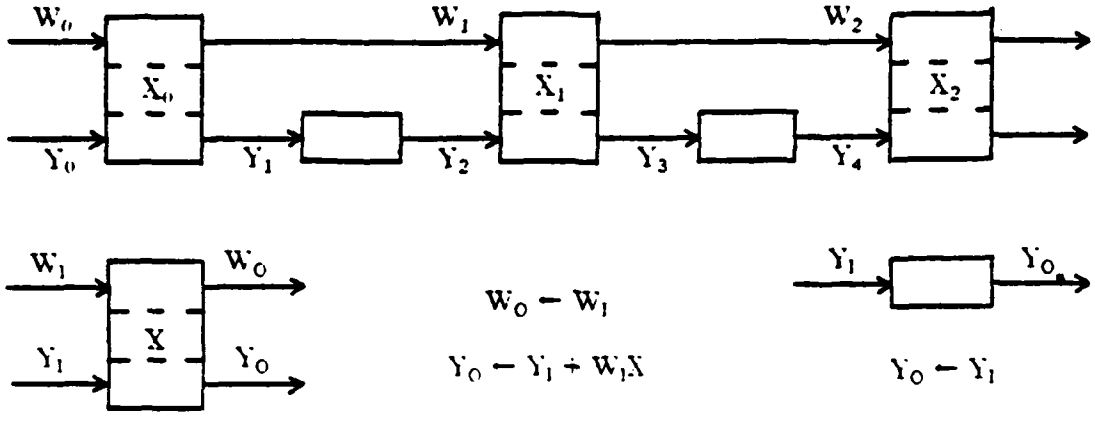



Figure 6. General structure of the +1 convolver and functional description of modules

Another canonical convolution network catalogued by Kung is R2 (results stay, inputs and weights move in same direction, but at different speeds). This design (Figure 7) has the following data flow parameters:

$$canonical(R2) \begin{cases} v_w = 1/2,\ v_x = 1,\ v_y = 0. \\ L_w = 1/2,\ L_x = -1,\ L_y = 1. \end{cases}$$

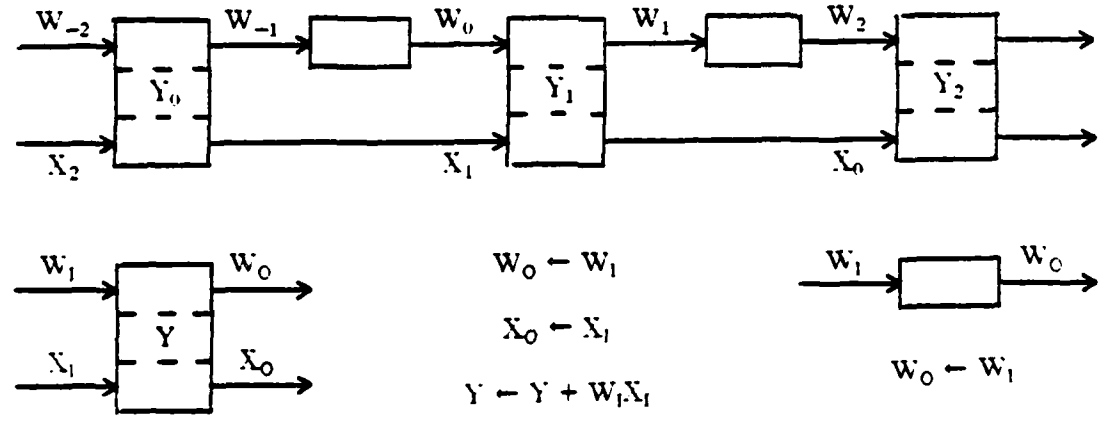



Figure 7. General structure of the canonical convolver (R2) and functional description of modules

Kung mentions that this design has a "dual" in which the weights move twice as fast as the inputs. This is clearly a result of the symmetry of convolution, which allows us to interchange the W and X data flow parameters; in fact, the W and X data flow parameters of any systolic convolution network

can be interchanged to yield another valid convolution network.

Now, simply subtracting $v_w$ from all data flow velocities in the canonical network yields a new convolution network in which the weights are stationary. The resulting data flow velocities are

$$-1/2 \mid v_w = 0, v_x = 1/2, v_y = -1/2.$$

This design is labeled W1 (weights stay, inputs and results move in opposite directions) by Kung (Figure 8).

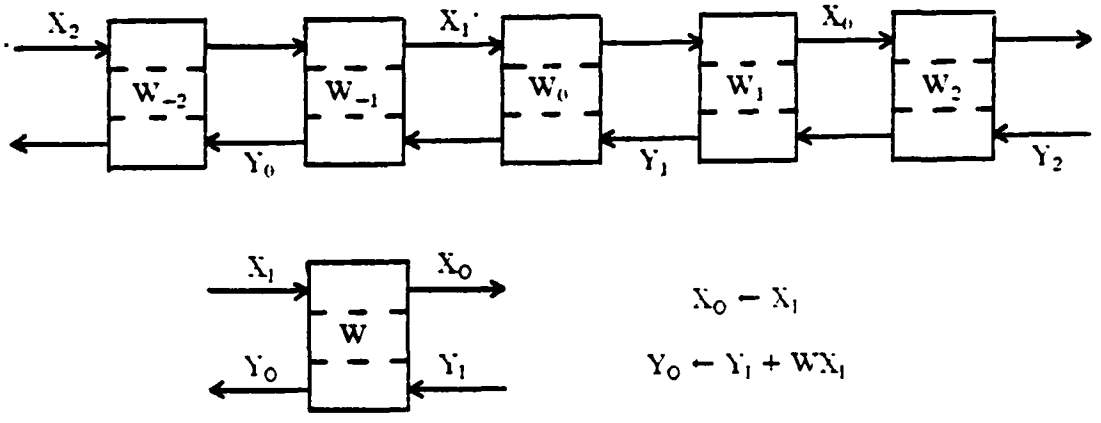



Figure 8. General structure of the -1/2 convolver and functional description of modules

Suppose that we subtract instead $v_x$ from the velocities of the canonical data flows. The resulting data flow velocities are

$$-1 \mid v_w = -1/2, v_x = 0, v_y = -1.$$

In this network (Figure 9), the inputs are stationary, and the weights and results move in the same direction but at different speeds. If we interchange the data flow parameters of W and X, we obtain the design labeled W2 (weights stay, inputs and results move in the same direction but at different speeds) by Kung.

We might also seek a transformation to demonstrate the equivalence of R1 and R2. However, the equivalence of R1 and R2 is not an immediate consequence of affine transformations or the symmetry of convolution. One can observe that the signs of $L_w$ and $L_x$ are the same for R1; for this reason, we

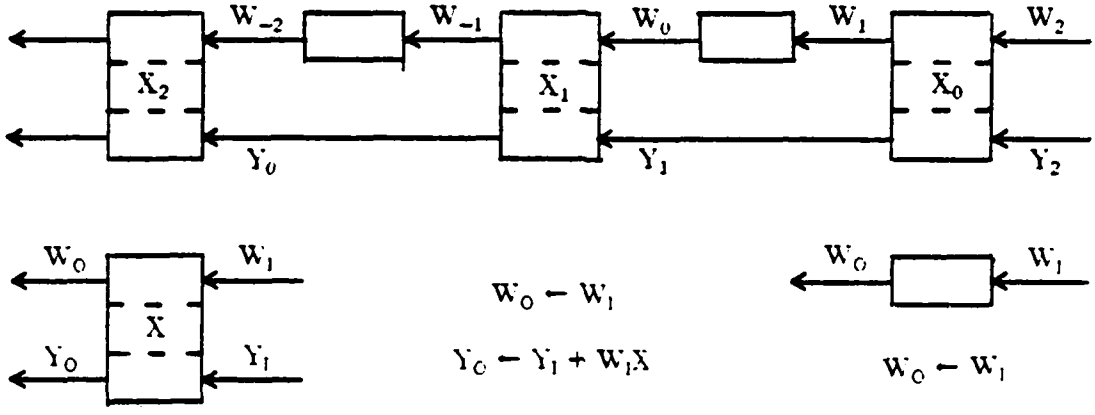



Figure 9. General structure of the -1 convolver and functional description of modules

say that R1 is a *cogradient* convolution network. For R2, though, the signs of $L_w$ and $L_x$ are opposite; for this reason, we say that R2 is a *contragradient* convolution network. Clearly, no affine transformation or simple interchange of data flow parameters will map R1 to R2. Thus, it seems that all of the known systolic convolution designs lie in two affine equivalence classes, the class of cogradient designs and the class of contragradient designs. These designs are summarized in the table below (Figure 10).

| affine class | linear class | $\nabla_w$ | $\nabla_x$ | $\nabla_y$ | Kung's label: unchanged | Kung's label: W⟷X |
|---|---|---|---|---|---|---|
| cogradient ($L_w$=2, $L_x$=2, $L_y$=1) | 0 | 1 | -1 | 0 | R1 | — |
| | -1 | 0 | -2 | -1 | "dual" of W2 | — |
| | +1 | 2 | 0 | 1 | — | — |
| contragradient ($L_w$=1/2, $L_x$=−1, $L_y$=1) | 0 | 1/2 | 1 | 0 | R2 | "dual" of R2 |
| | -1/2 | 0 | 1/2 | -1/2 | W1 | — |
| | -1 | -1/2 | 0 | -1 | — | W2 |

Figure 10. Summary of the systolic convolution networks

### 3.4.2. MATRIX MULTIPLICATION

Another problem for which systolic solutions have been proposed is that of matrix multiplication, i.e., given two matrices A and B, find the product matrix C = AB. The canonical systolic network for

matrix multiplication is the planar array of "orthogonally-connected" PEs (Figure 11). In [11], Preparata and Vuillemin describe this network (rotated -90°) and show that its operation may be viewed as a pipelined interaction of columns of A with rows of B. This pipelined interaction of columns with rows is a central feature of many systolic computations and will be further explored later. Since this network is a planar array, the unit grid becomes planar, the velocities of the data flows are two-component vectors, and the distortion matrices are 2×2 matrices.

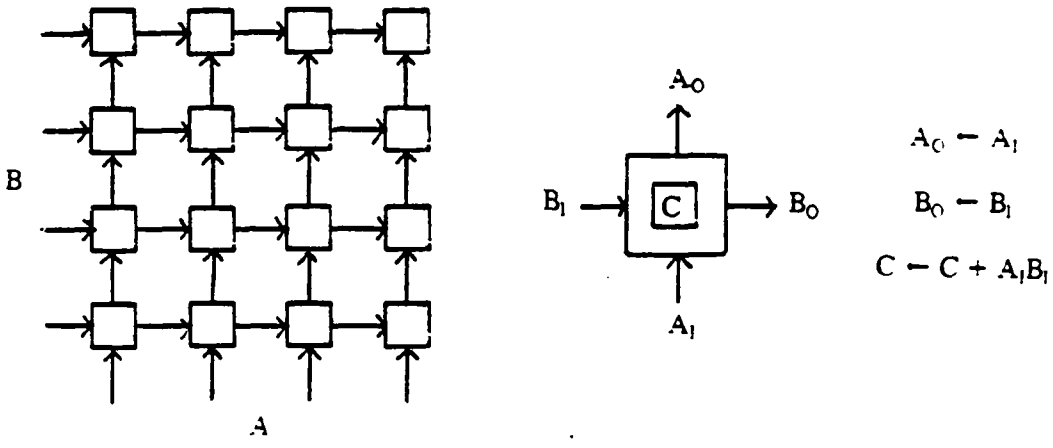



Figure 11. General structure of the canonical ($[0\ 0]^T$) matrix multiplier and functional description of modules

There are three data flows in this network. The northward data flow consists of the elements of A. The eastward data flow consists of the elements of B. The third data flow is stationary and consists of the elements of the product matrix, C, which are accumulated in the module registers. The parameters of these data flows will be subscripted with a, b, and c, respectively. In this thesis, we will assume that $\gamma$ is defined for all matrices such that $\gamma(a[i,j]) = [i\ \ j]^T$. The data flow parameters are

$$\mathbf{v}_a = [0\ \ 1]^T,\ \mathbf{v}_b = [1\ \ 0]^T,\ \mathbf{v}_c = [0\ \ 0]^T,$$

$$L_a = \begin{vmatrix} 1 & 0 \\ -1 & -1 \end{vmatrix},\ L_b = \begin{vmatrix} -1 & -1 \\ 0 & 1 \end{vmatrix},\ L_c = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}.$$

A number of potentially useful linear equivalence classes of networks for matrix multiplication may be derived by applying transformations to this network, as specified in Theorem 1. First, let us

consider the network obtained by adding the vector $[-1\ \ -1]^T$ to all velocities. The new velocities are (again, distortion matrices remain unchanged)

$$\mathbf{v}_a = [-1\ \ 0]^T,\ \mathbf{v}_b = [0\ \ -1]^T,\ \mathbf{v}_c = [-1\ \ -1]^T.$$

As in the case of convolution, all the active PEs in the original network are inner product step processors. Therefore, all the PEs in this network are also inner product step processors. Because each PE has six neighbors, the network (Figure 12) is said to be "hexagonally-connected."

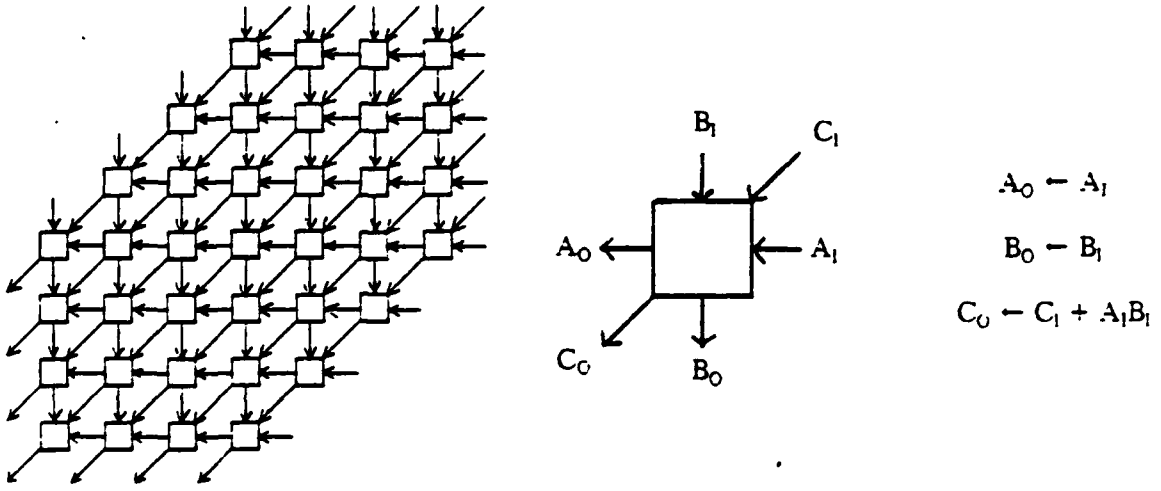



Figure 12. General structure of the $[-1\ \ -1]^T$ matrix multiplier

Now, if we add the vector $[-1/2\ \ -1/2]^T$ to all velocities in the canonical network, we obtain a network (Figure 13) that is again "orthogonally-connected," but communication along one of the axes is bidirectional now. The network has these data flow velocities:

$$\mathbf{v}_a = [-1/2\ \ 1/2]^T,\ \mathbf{v}_b = [1/2\ \ -1/2]^T,\ \mathbf{v}_c = [-1/2\ \ -1/2]^T.$$

Yet another network can be obtained by adding the vector $[-1/3\ \ -1/3]^T$ to all velocities in the canonical network. The resulting network (Figure 14) is "hexagonally-connected" with the following data flow velocities:

$$\mathbf{v}_a = [-1/3\ \ 2/3]^T,\ \mathbf{v}_b = [2/3\ \ -1/3]^T,\ \mathbf{v}_c = [-1/3\ \ -1/3]^T.$$

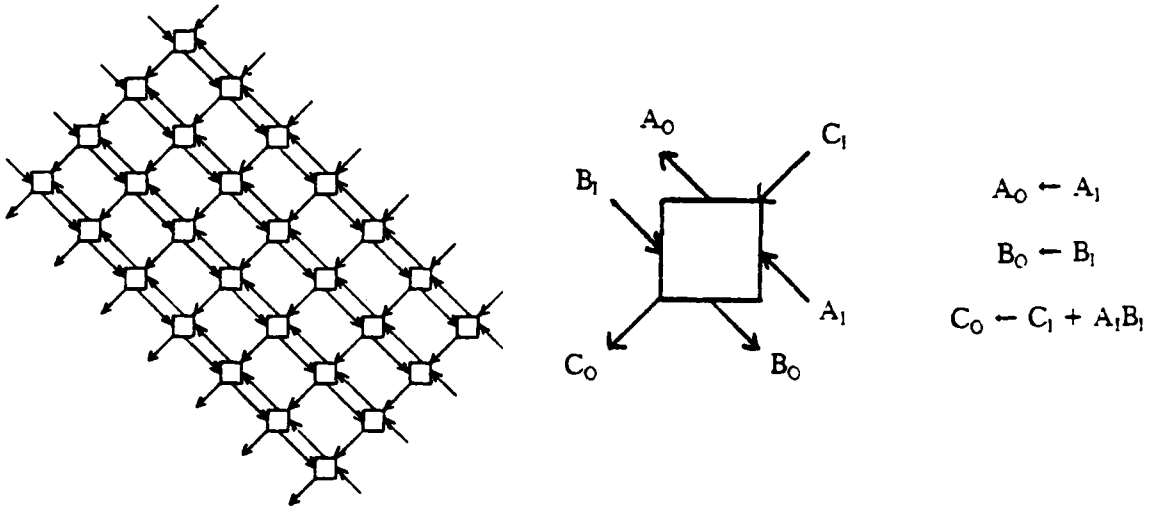



Figure 13. General structure of the $[-1/2 \ \ -1/2]^T$ matrix multiplier

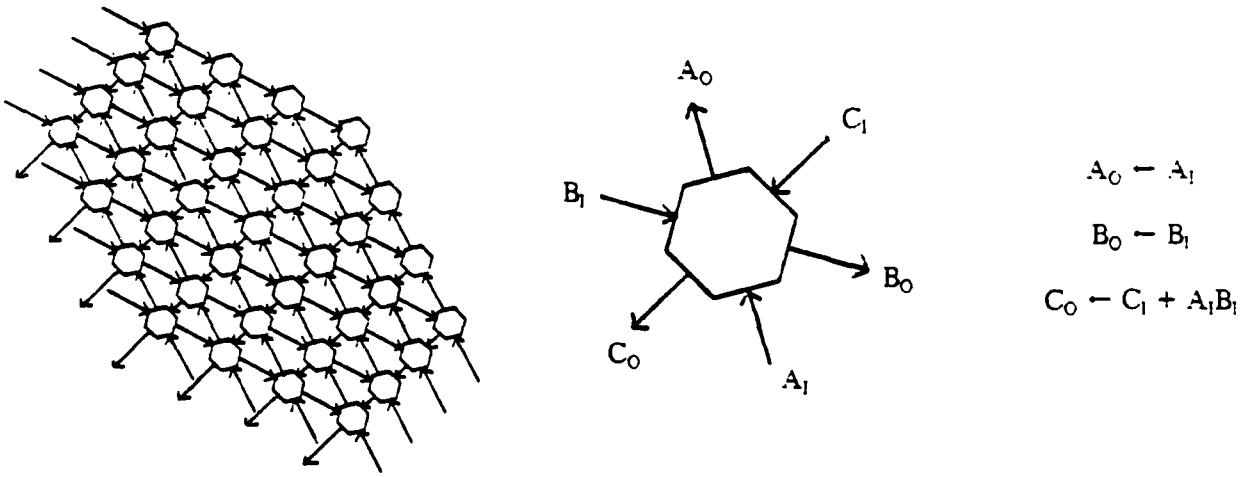



Figure 14. General structure of the $[-1/3 \ \ -1/3]^T$ matrix multiplier

It is important to note that each of the three matrix multiplication networks derived above is representative of a linear equivalence class of systolic networks: each can be "redrawn" in a more conventional or pleasing manner simply by multiplying the data flow parameters by the appropriate matrix. For instance, suppose we let

$$\mathbf{M} = \begin{vmatrix} -3/2 & 3/2 \\ -3 & -3 \end{vmatrix}.$$

If we multiply the data flow parameters of the $[-1/3 \;\; -1/3]^T$ network by **M**, we obtain the data flow parameters of the Kung-Leiserson systolic matrix multiplier [8]:

$$\mathbf{v}_a = [3/2 \;\; -1]^T, \;\; \mathbf{v}_b = [-3/2 \;\; -1]^T, \;\; \mathbf{v}_c = [0 \;\; 2]^T,$$

$$\mathbf{L}_a = \begin{vmatrix} -3 & -3/2 \\ 0 & 3 \end{vmatrix}, \;\; \mathbf{L}_b = \begin{vmatrix} 3/2 & 3 \\ 3 & 0 \end{vmatrix}, \;\; \mathbf{L}_c = \begin{vmatrix} -3/2 & 3/2 \\ -3 & -3 \end{vmatrix}.$$

Note that this network (Figure 15), when specialized to the case of banded matrices considered by Kung and Leiserson, has a parallelogram shape, as did their network.

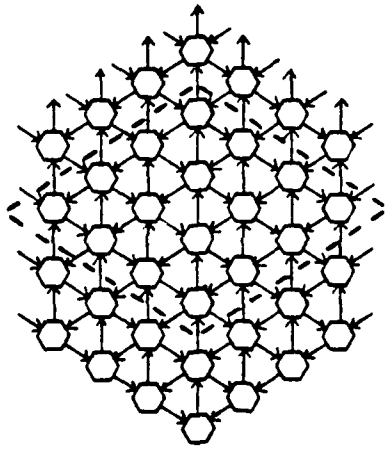


Figure 15. General structure of the Kung-Leiserson matrix multiplier (generalized to dense matrices)

### 3.4.3. LU DECOMPOSITION

The solution of systems of linear equations is another problem that has been approached with systolic techniques. This problem is usually posed in matrix form as follows: given a nonsingular n×n matrix A and an n×m matrix C, find the n×m matrix B such that C = AB. The solution of this problem, for general matrices, usually involves decomposing A into triangular factors and then solving triangular linear systems, both of which can be done directly with systolic designs. Kung and Leiserson have presented a systolic network for LU decomposition in [8]. This network (Figure 16) computes two

matrices, L and U, such that the input matrix, A, can be expressed as A = LU, where L is unit lower triangular and U is upper triangular.

In the Kung-Leiserson design, the matrix A flows northward into a network of "hexagonally-connected" processors. The L and U matrices may be retrieved from the network in a variety of ways; we will choose an implementation of the Kung-Leiserson design that more clearly exhibits network symmetry: L flows southeast from the lower-right margin of the network, and U flows southwest from the lower-left margin of the network. These three matrices comprise the data flows in the network. The corresponding data flow parameters will be subscripted with a, *l*, and u, respectively. Analysis of the network yields the following values for the data flow velocities and distortion matrices:

$$\mathbf{v}_l = [3/2 \;\; -1]^T, \;\; \mathbf{v}_u = [-3/2 \;\; -1]^T, \;\; \mathbf{v}_a = [0 \;\; 2]^T,$$

$$L_l = \begin{bmatrix} -3 & -3/2 \\ 0 & 3 \end{bmatrix}, \; L_u = \begin{bmatrix} 3/2 & 3 \\ 3 & 0 \end{bmatrix}, \; L_a = \begin{bmatrix} -3/2 & 3/2 \\ -3 & -3 \end{bmatrix}.$$

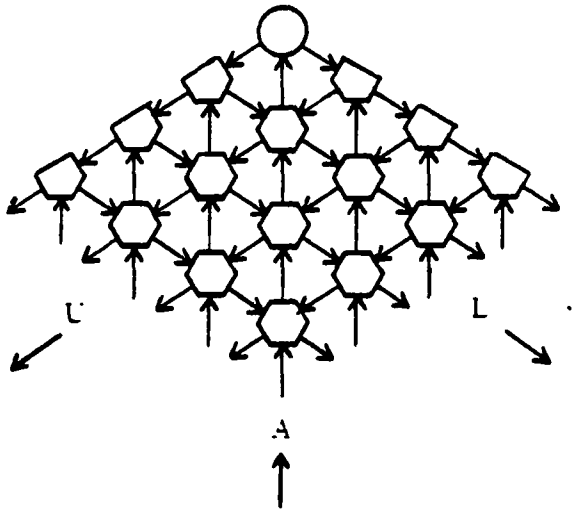



Figure 16. The Kung-Leiserson network for LU decomposition

There are four module types in this network: the PE at the top of the network is a T-module; the PEs at the upper-left margin (except the top) are S-modules; the PEs at the upper-right margin (except the top) are R-modules; all of the other PEs are G-modules. The module descriptions are given

below (Figure 17). The assignment statements are written in a concurrent form, as indicated by the use of commas to separate them.

| module type | state label | assignment statements | control statements |
|---|---|---|---|
| T (top) | T1: | L ← 1, U ← A | |
| S (u.-1.) | S1: | L ← A/U, U ← U | |
| R (u.-r.) | R1: | L ← L, U ← A | |
| G (others) | G1: | L ← L, U ← U, A ← A - LU | |

Figure 17. Module descriptions for the Kung-Leiserson LU decomposer

Now, we will derive a canonical network for LU decomposition. In this computation, the operation is matrix multiplication, the operands are the L and U matrices, and the result is the matrix A. The side conditions are that L is unit lower triangular and U is upper triangular. Therefore, the canonical design is the one in which the A data flow is stationary and has an identity distortion matrix. It is obtained, first, by adding $-\mathbf{v}_a = [0\ \ -2]^T$ to all data flow velocities and, second, by multiplying all data flow velocities and distortion matrices by $L_a^{-1}$. (Note that the Kung-Leiserson network is a member of the $L_a^{-1}\mathbf{v}_a = [-1/3\ \ -1/3]^T$ linear equivalence class.) The results of the transformation are as follows:

$$\mathbf{v}_l' = [0\ \ 1]^T,\ \mathbf{v}_u' = [1\ \ 0]^T,\ \mathbf{v}_a' = [0\ \ 0]^T,$$

$$L_l' = \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix},\ L_u' = \begin{bmatrix} -1 & -1 \\ 0 & 1 \end{bmatrix},\ L_a' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This canonical network is shown below (Figure 18).

We now must describe the functions of the PEs in the canonical network. In all of the previously discussed transformations, obtaining the module descriptions was trivial since the networks had only one module type and that module had a single state. In this case, however, the state flow technique must be employed. In the Kung-Leiserson network, the PEs do not change state, so the state flow has zero velocity ($\mathbf{v}_s = [0\ \ 0]^T$). If we take $G_s$ to be the convex grid set below (Figure 19), then

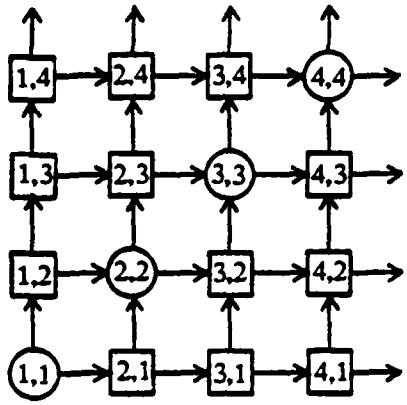



Figure 18. The canonical network for LU decomposition

$$L_1 = \begin{vmatrix} -3/2 & 3/2 \\ -1 & -1 \end{vmatrix}.$$

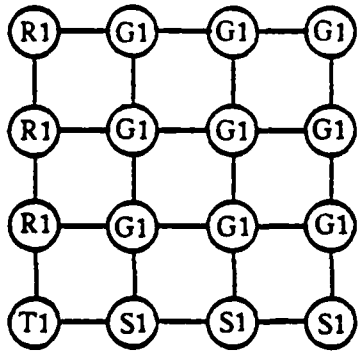



Figure 19. The convex grid set underlying the state flow

Transforming the state flow parameters as we did the data flow parameters, we obtain:

$$v_1' = L_2^{-1}(v_1 - v_2) = \begin{vmatrix} -1/3 & -1/6 \\ 1/3 & -1/6 \end{vmatrix} \times \left( \begin{vmatrix} 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 0 \\ 2 \end{vmatrix} \right) = \begin{vmatrix} 1/3 \\ 1/3 \end{vmatrix},$$

$$L_1' = L_2^{-1} L_1 = \begin{vmatrix} -1/3 & -1/6 \\ 1/3 & -1/6 \end{vmatrix} \times \begin{vmatrix} -3/2 & 3/2 \\ -1 & -1 \end{vmatrix} = \begin{vmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{vmatrix}.$$

We will derive the module descriptions for the canonical network from this transformed state flow (Figure 20).

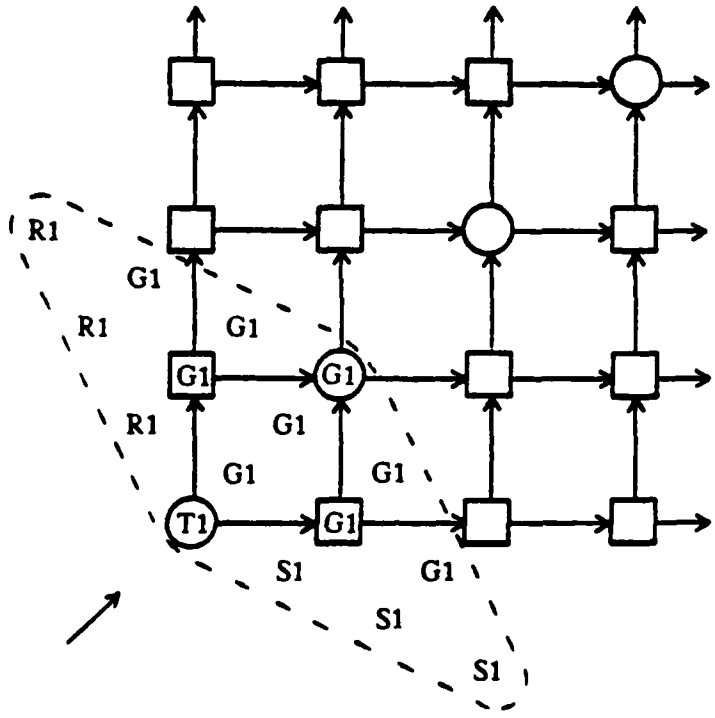



Figure 20. The state flow of the canonical LU decomposer
(at first clock cycle)

We can observe that the PEs on the diagonal of the network (those of the form M[i,i]) can only enter states G1 and T1. The PEs below the diagonal (those of the form M[i,j], $i > j$) can enter only G1 and S1, and the PEs above the diagonal (those of the form M[i,j], $i < j$) can enter only G1 and R1. This suggests three new module types: D-modules (M[i,i]) with states D1 = G1 and D2 = T1, L-modules (M[i,j], $i > j$) with states L1 = G1 and L2 = S1, and U-modules (M[i,j], $i < j$) with states U1 = G1 and U2 = R1. (Here, equality of states indicates computational equivalence, i.e., the assignment statements of the states are the same.)

We will now determine a hard-wired state transition scheme that realizes this state flow. Since the only part of the state flow that is crucial to the operation of the network is that which coincides with the data flows, we will find it convenient to assume that all PEs are initially in the state corresponding to G1, i.e., D-modules are in state D1, L-modules are in L1, and U-modules are in U1. This state acts as a quasi-quiescent state, in the sense that the register contents of a PE in this state are

not altered unless both the L and U data flows are flowing through the PE. The first PE to be excited from this state is M[1,1], which enters state D2. This must be accomplished via an external signal. After each M[i,i] enters this state, the processor immediately to the right of it enters L2, and the processor immediately above it enters U2. Therefore, we must include in the control statements of state D2 a device to indicate that these state transitions occur. We do this with "goto" control statements: a goto preceded by a PE reference denotes a state transition control signal to be sent to a neighboring PE; one without a PE reference denotes a state transition to be executed by the PE itself. So, for state D2, we include "M[i+1,i] goto L2" and "M[i,i+1] goto U2." The state L2 propagates to the east, so its control statements include "M[i+1,j] goto L2." Similarly, the state U2 propagates to the north, so its control statements include "M[i,j+1] goto U2." Immediately following these second states, the PEs may return to the quasi-quiescent state, so we include in the control statements of D2, L2, and U2 "goto D1," "goto L1," and "goto U1," respectively. The only remaining issue, then, is to determine a mechanism for D2 to propagate along the network diagonal. There are several possible solutions, however, the most straightforward of these is to include in the control statements of state D2 "M[i+1,i+1] goto D2 in 3." By this, we mean that M[i+1,i+1] is to transition to state D2 in three clock cycles. This can be achieved by buffering the control signal with a two-stage shift register. The module descriptions that finally emerge are shown below (Figure 21).

| module type | state label | assignment statements | control statements |
|---|---|---|---|
| D (M[i,i]) | D1:<br>D2: | L ← L, U ← U, A ← A - LU<br>L ← 1, U ← A | M[i+1,i] goto L2,<br>M[i,i+1] goto U2,<br>M[i+1,i+1] goto D2 in 3,<br>goto D1 |
| L (M[i,j], i > j) | L1:<br>L2: | L ← L, U ← U, A ← A - LU<br>L ← A/U, U ← U | M[i+1,j] goto L2,<br>goto L1 |
| U (M[i,j], i < j) | U1:<br>U2: | L ← L, U ← U, A ← A - LU<br>L ← L, U ← A | M[i,j+1] goto U2,<br>goto U1 |

Figure 21. Module descriptions for the canonical LU decomposer

### 3.4.4. TRIANGULAR SYSTEM SOLUTION

As was mentioned previously, the solution of triangular systems of linear equations is an important component in the problem of solving more general systems of linear equations. The lower-triangular variant of this problem is the following: given a nonsingular, lower-triangular matrix L and a matrix Y, find the matrix X such that Y = LX. In [8], Kung and Leiserson propose a systolic network for solving this problem in the special case where Y and X are column vectors. Here, we make minor modifications to their network and generalize it (the details are omitted for brevity) to obtain the network below (Figure 22), which solves lower-triangular linear systems in full generality (Y and X are matrices).

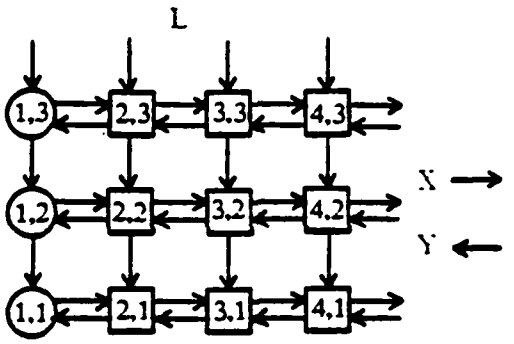



Figure 22. A systolic network for triangular system solution

The matrix L flows southward into the "orthogonally-connected" network of processors, Y flows westward into the network, and X flows eastward from the network. These, in fact, are the three data flows in the network; their parameters will be subscripted with *l*, y, and x, respectively. The values are

$$\mathbf{v}_l = [0 \;\; -1]^T, \;\; \mathbf{v}_x = [1 \;\; 0]^T, \;\; \mathbf{v}_y = [-1 \;\; 0]^T,$$

$$\mathbf{L}_l = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \;\; \mathbf{L}_x = \begin{bmatrix} -2 & -1 \\ 0 & -1 \end{bmatrix}, \;\; \mathbf{L}_y = \begin{bmatrix} 2 & 1 \\ 0 & -1 \end{bmatrix}.$$

There are two module types in this network: the PEs at the left margin are D-modules; the remaining ones are M-modules. The module descriptions are given below (Figure 23). Initially, all PEs are in a quiescent state, state 0. To initiate the computation, state D1 is externally excited in the

uppermost D-module (M[1,p]).

| module type | state label | assignment statements | control statements |
|---|---|---|---|
| D (M[1,j]) | D1: | L ← L, X ← Y/L | M[1,j-1] goto D1, M[2,j] goto M1, goto D2 |
| | D2: | L ← L, X ← Y/L | |
| M (M[i,j], i>1) | M1: | L ← L, X ← X, Y ← Y - LX | M[i+1,j] goto M1, goto M2 |
| | M2: | L ← L, X ← X, Y ← Y - LX | |

Figure 23. Module descriptions for the triangular system solver

To derive the canonical network, we observe that, for this computation, the operation is again matrix multiplication, the operands are the L and X matrices, and the result is the matrix Y. Therefore, we add $-\mathbf{v}_y = [1\ \ 0]^T$ to all data flow velocities and then multiply all data flow velocities and distortion matrices by $L_y^{-1}$. This implies that the original network is a representative of the $L_y^{-1}\mathbf{v}_y = [-1/2\ \ 0]^T$ linear equivalence class. The parameters that we obtain for the canonical network are

$$\mathbf{v}_l' = [0\ \ 1]^T,\ \mathbf{v}_x' = [1\ \ 0]^T,\ \mathbf{v}_y' = [0\ \ 0]^T,$$

$$L_l' = \begin{vmatrix} 1 & 0 \\ -1 & -1 \end{vmatrix},\ L_x' = \begin{vmatrix} -1 & -1 \\ 0 & 1 \end{vmatrix},\ L_{\cdot}' = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}.$$

Once again, the canonical network has an "orthogonally-connected" architecture (Figure 24).

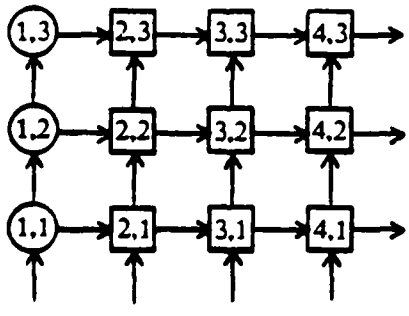



Figure 24. The canonical network for triangular system solution

Here too, we must employ the state flow technique to obtain the module descriptions for the canonical network. The first step is to obtain a state flow for the original $[-1/2\ \ 0]^T$ network. Observing that state D1 is first excited in M[1,p] and then propagates to M[1,p-1] and then to M[1,p-2] and so forth, suggests a state flow headed by D1 with velocity $[0\ \ -1]^T$. In fact, since D1 = D2 and M1 = M2, the state flow is essentially the same as the L data flow, with state D1 occupying the positions of the diagonal elements and state M1 occupying the other positions (Figure 25). If we choose $G_s$ as shown below (Figure 26), then we have these state flow parameters:

$$\mathbf{v}_s = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad L_s = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Performing the transformation on these parameters:

$$\mathbf{v}_s' = L_y^{-1}(\mathbf{v}_s - \mathbf{v}_y) = \begin{bmatrix} 1/2 & 1/2 \\ 0 & -1 \end{bmatrix} \times \left( \begin{bmatrix} 0 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$L_s' = L_y^{-1} L_s = \begin{bmatrix} 1/2 & 1/2 \\ 0 & -1 \end{bmatrix} \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix}.$$

We will derive the module descriptions for the canonical network from this transformed state flow (Figure 27).

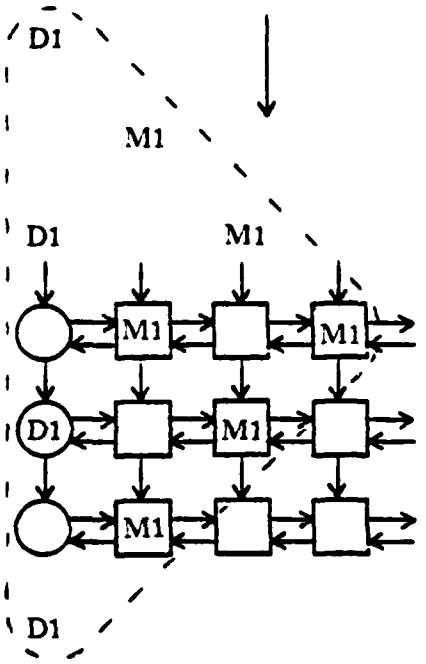



Figure 25. The state flow of the $[-1/2 \;\; 0]^T$ triangular system solver (at forth clock cycle)

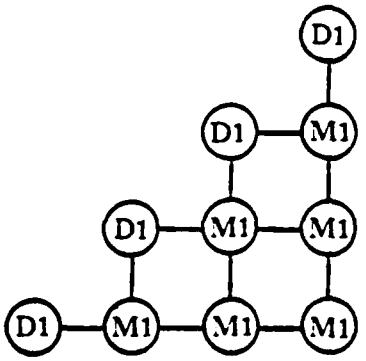



Figure 26. The convex grid set underlying the state flow

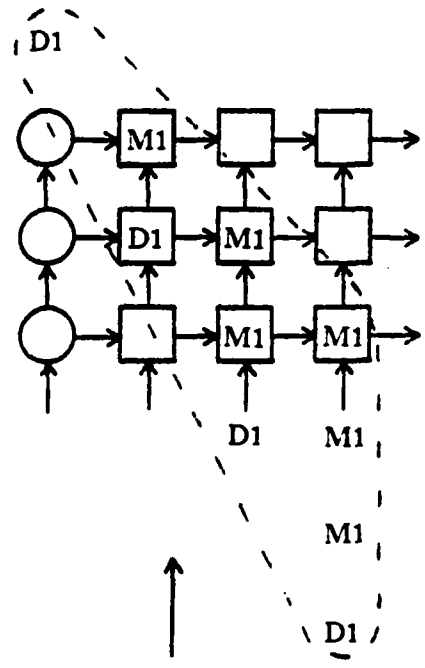



Figure 27. The state flow of the canonical triangular system solver
(at fourth clock cycle)

We now note that the PEs on the left margin of the network (those of the form M[1,j]) only enter state D1. All other PEs (those of the form M[i,j], i>1) can enter state D1 or M1. This suggests two new module types: A-modules (M[1,j]) with a single state A1 = D1 and B-modules (M[i,j], i>1) with states B1 = M1 and B2 = D1.

As before, we must determine a hard-wired state transition scheme that realizes this state flow. We assume that all PEs are initially in the quiescent state, state 0. The first PE to be excited from this state (via external signal) is M[1,1], which enters state A1. State A1 then propagates northward, so we include "M[1,j+1] goto A1" in the control statements of state A1. One cycle after each A-module enters state A1, the B-module immediately to its right enters state B1; two cycles after, the B-module enters state B2. Therefore, we also include "M[2,j] goto B1" and "M[2,j] goto B2 in 2" in the control statements of state A1. State B1 then propagates eastward, so we include "M[i+1,j] goto B1" in the control statements of state B1. State B2 also propagates eastward, however at a rate of 1/2 processor per cycle. Therefore, we also include "M[i+1,j] goto B2 in 2" in the control statements of state B2. All PEs are

appropriately returned to the quiescent state by including "goto 0" in the control statements of states A1 and B2. Thus, we derive the module descriptions below (Figure 28).

| module type | state label | assignment statements | control statements |
|---|---|---|---|
| A (M[1,j]) | A1: | L ← L, X ← Y/L | M[i,j+1] goto A1,<br>M[2,j] goto B1,<br>M[2,j] goto B2 in 2,<br>goto 0 |
| B (M[i,j], i>1) | B1:<br>B2: | L ← L, X ← X, Y ← Y - LX<br>L ← L, X ← Y/L | M[i+1,j] goto B1<br>M[i+1,j] goto B2 in 2,<br>goto 0 |

Figure 28. Module descriptions for the canonical triangular system solver

An interesting transformation of the canonical triangular system solver is one from which the resulting state flow has zero velocity. The networks with zero state flow velocity are those of the $-\mathbf{v}_t = [0\ \ -1]^T$ linear equivalence class and have the property that no PE changes state. These networks therefore have two module types, one corresponding to state A1 = B2 and one corresponding to state B1. Since no state transitions occur, the state descriptions of these modules do not contain any control statements, i.e., no control signals are transmitted through the network.

To exemplify such a network, we will add $-\mathbf{v}_t$ to all velocities and then multiply all velocities and distortion matrices by $\mathbf{M} = \mathbf{L}_t^{-1}$. The resulting parameters are

$$\mathbf{v}_l' = [0\ \ 0]^T,\ \mathbf{v}_x' = [1\ \ 0]^T,\ \mathbf{v}_y' = [0\ \ 1]^T,\ \mathbf{v}_t' = [0\ \ 0]^T,$$

$$\mathbf{L}_l' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \mathbf{L}_x' = \begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix},\ \mathbf{L}_y' = \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix},\ \mathbf{L}_t' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The resulting network is shown (Figure 29), and the module descriptions are given (Figure 30). Again, we note that the choice of M does not alter the topology of the network; it is simply chosen so that the network can be drawn in a more convenient manner (in this case, with the same layout as the underlying convex grid set, $G_t$).

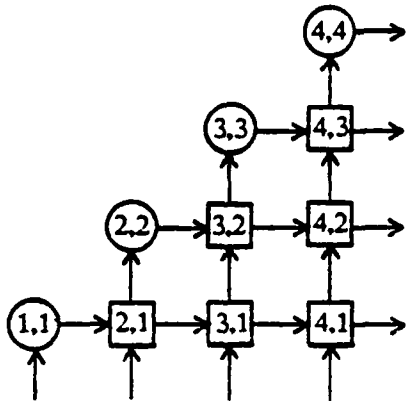



Figure 29. A $[0\ -1]^T$ network for triangular system solution

| module type | state label | assignment statements | control statements |
|---|---|---|---|
| E (M[i,i]) | E1: | L ← L, X ← Y/L | |
| F (M[i,j], i>j) | F1: | L ← L, X ← X, Y ← Y -LX | |

Figure 30. Module descriptions for the $[0\ 1]^T$ triangular system solver

The previous discussion of systolic triangular system solvers was pertinent to the lower-triangular variant of the problem. The following trivial observation, however, allows us to apply these results to the solution of upper-triangular systems. If U is a nonsingular, upper-triangular matrix, then solving the linear system W = UV can be reduced to solving the lower-triangular system Y = LX, where $L = RUR^{-1}$, Y = RW, X = RV, and R is a reverse permutation matrix, i.e.,

$$R = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \cdot & \cdots & 1 & 0 \\ \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdots & \cdot & \cdot \\ 0 & 1 & \cdots & \cdot \\ 1 & 0 & \cdots & 0 \end{bmatrix}.$$

Another modification of this problem is to set Y = I (by hardwiring, perhaps) in order to handle an important special case, triangular matrix inversion. One such systolic network (for upper-triangular matrices) has been proposed by Preparata and Vuillemin [10].

## 3.5. CROSSINGS IN TRANSFORMED NETWORKS

### 3.5.1. NECESSARY AND SUFFICIENT CONDITIONS FOR CROSSING

As one experiments with these transformations on two-dimensional systolic networks, one may observe that many of them result in networks with communication edges that cross. Formally, a *crossing* is the intersection of two nonparallel communication edges that have distinct endpoints. For instance, if we transform the canonical matrix multiplier by adding the vector $[-1/4\ \ -1/4]^T$ to all data flow velocities, we obtain a systolic network with crossings (Figure 31). Networks with such crossings are somewhat undesirable since they are no longer planarly embeddable in the grid. This provides the motivation to characterize the conditions that cause crossing.

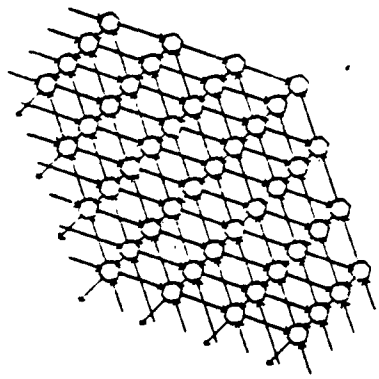


Figure 31. A $[-1/4\ \ -1/4]^T$ matrix multiplier

We now establish a necessary condition for crossings in a systolic network.

**Lemma 1:**

In a *connected* systolic network (one in which the underlying undirected graph is connected) with m data flows having velocities $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m$, a crossing exists only if there exists an m-component vector, $\mathbf{x}$, in the null space of $V = [\mathbf{v}_1\ \cdots\ \mathbf{v}_m]$ with exactly one or two noninteger components corresponding to nonzero, linearly independent columns of V.

**Proof:** Consider two edges e, $(\mathbf{p}_e, \mathbf{p}_e + \mathbf{v}_i)$, and f, $(\mathbf{p}_f, \mathbf{p}_f + \mathbf{v}_j)$, that cross at a point $\mathbf{p}$. We

then have the following:

$$\mathbf{p}_e + \xi_e \mathbf{v}_i = \mathbf{p}_f + \xi_f \mathbf{v}_j = \mathbf{p}, \text{ where } \xi_e, \xi_f \in [0,1]$$

$$\mathbf{p}_e - \mathbf{p}_f + \xi_e \mathbf{v}_i - \xi_f \mathbf{v}_j = 0$$

Because $\mathbf{p}_e$ and $\mathbf{p}_f$ are both vertices of the systolic network, and because the systolic network is connected, some sequence of edges (some possibly traversed in the reverse direction) will form a path from $\mathbf{p}_e$ to $\mathbf{p}_f$. Therefore, $\mathbf{p}_e - \mathbf{p}_f = \mathbf{V}\mathbf{x}'$, where $\mathbf{x}'$ is a vector of integers and

$$\mathbf{V}\mathbf{x}' + \xi_e \mathbf{v}_i - \xi_f \mathbf{v}_j = 0.$$

If we let $\mathbf{x} = \mathbf{x}' + \xi_e \mathbf{e}_i - \xi_f \mathbf{e}_j$ where $\mathbf{e}_k$ is the unit vector with the kth coordinate equal to 1, then $\mathbf{V}\mathbf{x} = 0$, i.e., $\mathbf{x}$ is in the null space of $\mathbf{V}$. Since e and f have distinct endpoints, $\xi_e \in (0,1)$ or $\xi_f \in (0,1)$, so $\mathbf{x}$ has one or two noninteger components. If $\mathbf{x}$ has one noninteger component, that component must not correspond to a zero column of $\mathbf{V}$, otherwise, the distinct endpoint condition is violated. If $\mathbf{x}$ has two noninteger components, then they must not correspond to linearly dependent columns of $\mathbf{V}$ since, by definition, a crossing cannot occur between two parallel edges. □

We can also establish a sufficient condition for crossings in a systolic network.

**Lemma 2:**

In a systolic network in which each nonboundary PE has communication edges corresponding to every data flow, a crossing exists if there exists an m-component vector, $\mathbf{x}$, in the null space of $\mathbf{V}$ with exactly one or two noninteger components corresponding to nonzero, linearly independent columns of $\mathbf{V}$.

**Proof:** Suppose that $\mathbf{x}$ is a vector with two noninteger components, $x_i$ and $x_j$, satisfying the above criteria. Let $\xi_e = x_i - \lfloor x_i \rfloor$, $\xi_f = \lceil x_j \rceil - x_j$, and $\mathbf{x}' = \mathbf{x} - \xi_e \mathbf{e}_i + \xi_f \mathbf{e}_j$; $\mathbf{x}'$ is therefore a vector of integers. Let $\mathbf{p}_e$ be the position of an arbitrary nonboundary PE such that $\mathbf{p}_f = \mathbf{p}_e - \mathbf{V}\mathbf{x}'$ lies within the boundaries of the systolic network. Since each nonboundary PE has communication edges corresponding to every data flow, a simple inductive argument

shows that any integer linear combination of the data flow velocities, when added to $\mathbf{p}_e$, yields the position of another PE (as long as this combination lies within the boundaries of the network). In particular, then, $\mathbf{p}_f$ is also the position of a nonboundary PE. The following argument shows that edge e, $(\mathbf{p}_e, \mathbf{p}_e + \mathbf{v}_i)$, must cross edge f, $(\mathbf{p}_f, \mathbf{p}_f + \mathbf{v}_j)$:

$$\mathbf{p}_f = \mathbf{p}_e - \mathbf{V}\mathbf{x}'$$

$$\mathbf{p}_f = \mathbf{p}_e - \mathbf{V}(\mathbf{x} - \xi_e\mathbf{e}_i + \xi_f\mathbf{e}_j)$$

$$\mathbf{p}_f = \mathbf{p}_e - \mathbf{V}\mathbf{x} + \xi_e\mathbf{V}\mathbf{e}_i - \xi_f\mathbf{V}\mathbf{e}_j$$

$$\mathbf{p}_f = \mathbf{p}_e + \xi_e\mathbf{v}_i - \xi_f\mathbf{v}_j$$

$$\mathbf{p}_f + \xi_f\mathbf{v}_j = \mathbf{p}_e + \xi_e\mathbf{v}_i$$

A similar argument can be applied when $\mathbf{x}$ has only one noninteger component. □

Combining these two results in a trivial manner yields the following theorem.

Theorem 3:

In a connected systolic network in which each nonboundary PE has communication edges corresponding to every data flow, a crossing exists if and only if there exists an m component vector, $\mathbf{x}$, in the null space of $\mathbf{V}$ with exactly one or two noninteger components corresponding to nonzero, linearly independent columns of $\mathbf{V}$.

Now, we will utilize this result to study the affine transformations described earlier and their effect on crossing in systolic networks.

### 3.5.2. THE EFFECT OF AFFINE TRANSFORMATIONS ON CROSSING

First, let us examine the effect of Theorem 1 transformations on crossing. Suppose we are given a crossing-free systolic network with $\mathbf{V}$ as its matrix of data flow velocities. Let $\mathbf{x}$ be a vector with $S_x \triangleq \sum_k x_k \neq 0$ and two noninteger components, $x_i$ and $x_j$. We can select a $\mathbf{u}$ that induces crossings when added to all data flow velocities in the following manner. Since the original network is crossing-free, $\mathbf{V}\mathbf{x} \neq 0$. If $\mathbf{V}'$ represents the transformed matrix of data flow velocities, then $\mathbf{V}' = \mathbf{V} + \mathbf{U}$, where $\mathbf{U} = [\mathbf{u}\ \mathbf{u}\ \cdots\ \mathbf{u}]$.

$$\mathbf{V'x} = (\mathbf{V} + \mathbf{U})\mathbf{x}$$

$$\mathbf{V'x} = \mathbf{Vx} + \mathbf{Ux}$$

$$\mathbf{V'x} = \mathbf{Vx} + \mathbf{S_x u}$$

Therefore, if we choose $\mathbf{u} = -\mathbf{Vx}/\mathbf{S_x}$, then $\mathbf{V'x} = 0$. This implies that a crossing exists unless $\mathbf{v}_i'$ is parallel to $\mathbf{v}_j'$, which, in general, is not the case. This demonstrates the fact that a crossing-free systolic network may be transformed to one with crossings according to the rules of Theorem 1. Since this transformation is invertible, it also follows that a systolic network with crossings may be transformed to a crossing-free one. Thus, crossing-freedom is variant with respect to Theorem 1 transformations (which was expected since our example of a systolic network with crossings was derived by this type of transformation on a crossing-free systolic network).

Now, we will examine the effect of Theorem 2 transformations on crossing. Let $\mathbf{V}$ be the matrix of data flow velocities of the original network, and $\mathbf{V'} = \mathbf{MV}$ be the matrix of data flow velocities of the resulting network. If $\mathbf{Vx} = 0$, then $\mathbf{V'x} = \mathbf{MVx} = \mathbf{M0} = 0$. If $\mathbf{Vx} = \mathbf{w} \neq 0$, then $\mathbf{V'x} = \mathbf{MVx} = \mathbf{Mw} \neq 0$ since $\mathbf{M}$ is nonsingular (according to Theorem 2). Thus, crossing-freedom is invariant with respect to Theorem 2 transformations.

As an example, suppose we apply Theorem 3 to a suitably restricted systolic network with the following matrix of data flow velocities:

$$\mathbf{V} = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{vmatrix}$$ (for instance, the canonical matrix multiplier).

If $\mathbf{Vx} = 0$, then $x_1 = 0$, $x_2 = 0$, and $x_3$ is unconstrained, so we may choose $x_3$ to be noninteger. This component, however, corresponds to $\mathbf{v}_3$, which is a zero column of $\mathbf{V}$. Thus, no $\mathbf{x}$ can be chosen to satisfy the criteria of Theorem 3, and the network is crossing-free, as is evident (Figure 7). Now, let us transform this network by adding $\mathbf{u} = [-3/2 \;\; -1/2]^T$ to all data flow velocities:

$$\mathbf{V'} = \begin{vmatrix} -3/2 & -1/2 & -3/2 \\ 1/2 & -1/2 & -1/2 \end{vmatrix}.$$

If $\mathbf{V'x} = 0$, then $x_1 = (1/3)x_2 = (-1/2)x_3$. Therefore, $\mathbf{x} = [1/3 \;\; 1 \;\; -2/3]^T$ satisfies the criteria of

Theorem 3, and the transformation results in crossings (Figure 32).

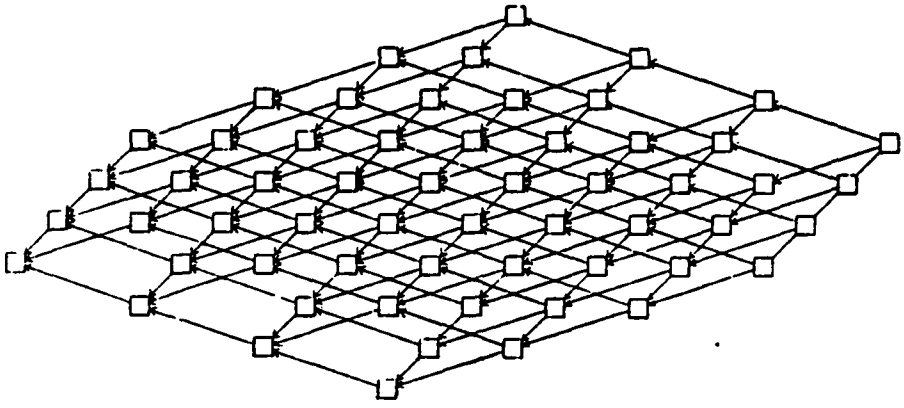


Figure 32. A $[-3/2 \quad -1/2]^T$ matrix multiplier (showing crossings)

### 3.5.3. ENUMERATION OF CROSSING-FREE CANONICAL CLASSES

The previous discussion leads us naturally to ask which of the linear equivalence classes of matrix multipliers is crossing-free. When we perform a Theorem 1 transformation on the canonical design, we obtain a new matrix of data flow velocities, $V' = V + U$, i.e.,

$$V' = \begin{bmatrix} u_1 & 1+u_1 & u_1 \\ 1+u_2 & u_2 & u_2 \end{bmatrix}.$$

From Theorem 3, we know that crossings will exist if and only if there exists a vector in the null space of $V'$ with one or two noninteger components. The null space of $V'$ is a one-dimensional subspace of $\mathbb{R}^3$ as long as $V'$ has full rank, which must be the case. Therefore, the null space of $V'$ can be represented as the range space of a 3×1 matrix, W, where $V'W = 0$. If we partition $V'$ into $[V'_1 \; V'_2]$, where $V'_1 = v_1'$ and $V'_2 = [v_2' \; v_3']$, we can rewrite this equation in the following way:

$$V'W = [V'_1 V'_2] \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = V'_1 W_1 + V'_2 W_2 = 0.$$

If $v_2'$ and $v_3'$ are linearly independent, then we can choose a form for W with $W_1 = 1$. Then we must have $W_2 = -(V'_2)^{-1} V'_1$. If $W_2$ contains a noninteger, then $x = W$ clearly satisfies the criteria of Theorem 3, and a crossing must exist. Otherwise, let $r = \max\{|w_2|, |w_3|\}$, i.e., r is the maximum

magnitude of the elements of $W_2$. If $r > 1$, then $x \approx W$ satisfies the criteria of Theorem 3 and a crossing must exist. The only remaining choices for $W_2$ are the $2\times1$ matrices over $\{-1, 0, +1\}$; these are $[0\ 0]^T$, $\pm[0\ 1]^T$, $\pm[1\ 0]^T$, $\pm[1\ 1]^T$, and $\pm[1\ -1]^T$.

We will now examine each of these possibilities on a case-by-case basis.

$$W_2 = -(V'_2)^{-1}V'_1 = -\begin{vmatrix} 1+u_1 & u_1 \\ u_2 & u_2 \end{vmatrix}^{-1} \begin{vmatrix} u_1 \\ 1+u_2 \end{vmatrix}$$

$$w_2 = \frac{u_1}{u_2},\quad w_3 = \frac{-(1+u_1+u_2)}{u_2}$$

Solving for $u$,

$$u_1 = \frac{-w_2}{(1+w_2+w_3)},\quad u_2 = \frac{-1}{(1+w_2+w_3)}.$$

$W_2 = [0\ 0]^T$: $u = [0\ -1]^T$

$W_2 = [0\ 1]^T$: $u = [0\ -1/2]^T$

$W_2 = [0\ -1]^T$: $u$ is undefined for this choice

$W_2 = [1\ 0]^T$: $u = [-1/2\ -1/2]^T$

$W_2 = [-1\ 0]^T$: $u$ is undefined for this choice

$W_2 = [1\ 1]^T$: $u = [-1/3\ -1/3]^T$

$W_2 = [-1\ -1]^T$: $u = [-1\ 1]^T$

$W_2 = [1\ -1]^T$: $u = [-1\ -1]^T$

$W_2 = [-1\ 1]^T$: $u = [1\ -1]^T$

Thus, there are seven linear equivalence classes of crossing-free matrix multipliers with $v_2'$ and $v_3'$ linearly independent.

We can apply the same argument as above when $v_1'$ and $v_3'$ are linearly independent simply by permuting the columns of $V'$ (first and second are interchanged). Proceeding in this manner, we obtain for the same possible choices of $W_2$:

$$u_1 = \frac{-1}{(1+w_2+w_3)}, \quad u_2 = \frac{-w_2}{(1+w_2+w_3)}.$$

Therefore, the set of vectors we obtain is the same as the set above except that the first and second components are interchanged. This actually yields only two new vectors, $[-1 \;\; 0]^T$ and $[-1/2 \;\; 0]^T$.

We have considered all possible cases except that in which $v_2'$ and $v_3'$ are linearly dependent and $v_1'$ and $v_3'$ are linearly dependent. Since V must have full rank, this can be true only if $v_3' = 0 = v_3$. Indeed, this corresponds to the canonical network. Therefore, all together, there are ten linear equivalence classes of systolic matrix multipliers.

### 3.5.4. TWO DIMENSIONAL NETWORKS WITH FOUR OR MORE DATA FLOWS

Theorem 3 can also be used to show that certain broad classes of two-dimensional systolic networks must have crossings. The following theorem summarizes this interesting result.

**Theorem 4:**

A two-dimensional systolic network (restricted as per Theorem 3) with four or more pairwise linearly independent data flow velocities must exhibit crossings.

**Proof:** Let us assume that $v_1$, $v_2$, $v_3$, and $v_4$ are pairwise linearly independent. Since the crossings found by considering these four data flows alone are a subset of all the crossings in the network, we can, without loss of generality, consider the case of exactly four data flows:

$$V = [v_1 \; v_2 \; v_3 v_4] = \begin{vmatrix} v_{11} & v_{12} & v_{13} & v_{14} \\ v_{21} & v_{22} & v_{23} & v_{24} \end{vmatrix}.$$

We know from Theorem 3 that if a four-component vector, x, in the null space of V with exactly one or two noninteger components can be found, then crossings must exist in the network (since we already know that the columns of V are nonzero and nonparallel).

The null space of V is a two-dimensional subspace of $\mathbb{R}^4$ and, as such, can be characterized as the range space of a 4×2 matrix, W, where $VW = 0$. We will rewrite this equation in the following way:

$$\mathbf{VW} = [\mathbf{V}_1\ \mathbf{V}_2]\begin{bmatrix}\mathbf{W}_1\\ \mathbf{W}_2\end{bmatrix} = \mathbf{V}_1\mathbf{W}_1 + \mathbf{V}_2\mathbf{W}_2 = 0, \text{ where } \mathbf{V}_1 = [\mathbf{v}_1\ \mathbf{v}_2] \text{ and } \mathbf{V}_2 = [\mathbf{v}_3\ \mathbf{v}_4].$$

As before, we can let $\mathbf{W}_1 = \mathbf{I}$ and $\mathbf{W}_2 = -\mathbf{V}_2^{-1}\mathbf{V}_1$. Since the columns of $\mathbf{V}$ are pairwise linearly independent, $\mathbf{V}_1$ and $\mathbf{V}_2$ have full rank. Therefore, $\mathbf{W}_2$ exists and has full rank. Thus, all vectors in the null space of $\mathbf{V}$ have the form $\mathbf{x} = \mathbf{W}\mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^2$.

Arguing as we did in enumerating crossing-free matrix multipliers, we restrict the choices for $\mathbf{W}_2$ to those 2×2 matrices over {-1, 0, +1}. We can eliminate those containing 0 since the corresponding column of $\mathbf{W}$ would contradict the pairwise linear independence hypothesis. The only 2×2 matrices over {-1, +1} not eliminated by the fact that $\mathbf{W}_2$ has full rank are

$$\pm\begin{bmatrix}-1 & -1\\ -1 & +1\end{bmatrix},\ \pm\begin{bmatrix}-1 & -1\\ +1 & -1\end{bmatrix},\ \pm\begin{bmatrix}-1 & +1\\ -1 & -1\end{bmatrix}, \text{ and } \pm\begin{bmatrix}-1 & +1\\ +1 & +1\end{bmatrix}.$$

In each of these cases, $\mathbf{x} = \mathbf{W}[1/2\ \ 1/2]^T$ satisfies the conditions of Theorem 3. Thus, in all possible cases, some $\mathbf{x}$ in the null space of $\mathbf{V}$ can be found with exactly one or two noninteger components, and a crossing must exist. □

It is important to note, however, that if we remove the restriction that all nonboundary PEs must have communication edges corresponding to every data flow, we can construct crossing-free systolic networks with four pairwise linearly independent data flow velocities (Figure 33).

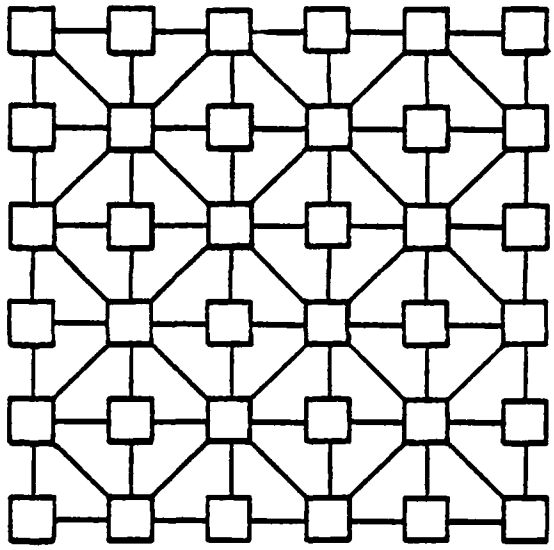


Figure 33. A crossing-free systolic network with four pairwise linearly independent data flow velocities

# 4. CONCLUSION

In this thesis, we have abstracted the parameters that we feel are important in specifying a systolic design. We have given a simple set of rules for transforming these parameters while preserving the underlying computation. In addition, we have shown how to derive a description of the processors in the resulting design through a state flow analysis. Finally, we have characterized those transformations that avoid the phenomenon of crossing.

We feel that these transformations may be useful to the designer implementing a systolic array as a means of tailoring the array to a specific application, e.g., to avoid preloading data registers (make all data flows have nonzero velocity), to ensure that processors need not change state (make the state flow have zero velocity), to make the array more compatible with solving partitioned problems of larger size (guarantee that no subset of the data flows forms a cycle), to make the array pipelinable (add an extra component to all parameters and add a velocity which is nonzero in this component). Furthermore, these transformations may prevent researchers from "reinventing" equivalent systolic arrays in an ad hoc manner. Perhaps the most interesting topic for further research would be a characterization of those problems or algorithms which are amenable to systolic processing. This elusive goal would further unify the theory of systolic arrays.

## REFERENCES


[1] Ahmed, H. M., Delosme, J.-M., and Morf, M., "Highly Concurrent Computing Structures for Digital Signal Processing and Matrix Arithmetic," *Computer Magazine*, Volume 15, Number 1, January 1982, pp. 65-82.

[2] Brent, R. P. and Kung, H. T., "Systolic VLSI Arrays for Polynomial GCD Computation," *IEEE Transactions on Computers*, Volume C-33, Number 8, August 1984, pp. 731-736.

[3] Brent, R. P. and Luk, F. T., "Computing the Cholesky Factorization Using a Systolic Architecture," Australian National University Department of Computer Science Technical Report, TR-CS-82-08, Canberra, Australia, August 1982.

[4] Brent, R. P. and Luk, F. T., "A Systolic Array for the Linear-Time Solution of Toeplitz Systems of Equations," Australian National University Department of Computer Science Technical Report, TR-CS-83-02, Canberra, Australia, January 1983.

[5] Cappello, P. R. and Steiglitz, K., "Unifying VLSI Array Design with Linear Transformations of Space-Time," *Advances in Computing Research*, Volume 2, VLSI Theory, (F. P. Preparata, Ed.), 1984, (to appear).

[6] Guibas, L. J., Kung, H. T., and Thompson, C. D., "Direct VLSI Implementation of Combinatorial Algorithms," *Proceedings Conference on Very Large Scale Integration: Architecture, Design, Fabrication*, California Institute of Technology, January 1979, pp. 509-525.

[7] Kung, H. T., "Why Systolic Architectures?" *Computer Magazine*, Volume 15, Number 1, January 1982, pp. 37-46.

[8] Kung, H. T. and Leiserson, C. E., "Systolic Arrays (for VLSI)," *Symposium on Sparse Matrix Computations*, Knoxville, Tennessee, November 1978, pp. 256-282.

[9] Leiserson, C. E. and Saxe, J. B., "Optimizing Synchronous Systems," *Proceedings 22nd IEEE Symposium on Foundations of Computer Science*, Nashville, Tennessee, October 1981, pp 23-36.

[10] Preparata, F. P. and Vuillemin, J. E., "Optimal Integrated Circuit Implementation of Triangular Matrix Inversion," *1980 International Conference on Parallel Processing*, Boyne, Michigan, August 1980, pp. 211-216.

[11] Preparata, F. P. and Vuillemin, J. E., "Area-Time Optimal VLSI Networks for Multiplying Matrices," *Information Processing Letters*, Volume 11, Number 2, October 1980, pp. 77-80.

END

FILMED

1-86

DTIC